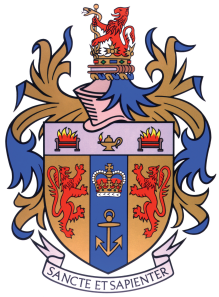
A formal approach for the analysis of the security of socio-technical systems

Sempreboni, Diego

Awarding institution:
King's College London

A formal approach for the analysis of the security of socio-technical systems



Diego Sempreboni

Supervisor: Prof. Luca Viganò

Department of Informatics

King's College London

This dissertation is submitted for the degree of

Doctor of Philosophy

September 2020

I would like to dedicate this thesis
to my beloved mother Caterina
and my beloved father Silvano.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

Diego Sempreboni

September 2020

Acknowledgements

Once, I heard this phrase:

> "Pursuing a PhD is not only an intellectual challenge, it is also a substantial emotional challenge".

A long journey.

Almost four years passed since the day I started and, I was not able to find better words to describe what I experienced than "Your journey was like a roller coaster". A friend of mine said this. She was right.

Along this journey, I met people. I had the privilege to share this journey with them. I had the privilege to share all of this with new but also old friends.
I want to thank all of them. I need to.

I would like to say thank you to my mother *Caterina Sarti* and my father *Silvano Sempreboni*, for supporting me in all possible ways they knew. People say that kids are gifts for parents. However, parents like them are a precious gift for someone like me.

I would like to thank my supervisor, *Prof. Luca Viganò*, a second father, a friend. I want to thank him for the many times he encouraged me, for our many fruitful and inspiring conversations, for our many "movie and burger time". I wouldn't be here writing these words if it was not for him.

I am also very thankful to my colleagues at King's. We shared tons of lunches, dinners, tea(r)s and more. I want to specially recognise the help of some of them. *Dr. Jonathan Cardoso Silva*, who helped me during the beginning of my journey in London when I was disoriented, *Ms. Francesca Mosca*, who was the friend I needed but that I never expected, *Ms. Mai Alzamel*, who shared with me coffees, smiles and values. Also, two friends are worth to be mentioned, *Dr. Gerard Canal Camprodon*, because even if we argued sometimes, you still remain a great friend of mine and *Dr. Xavier Ferrer Aran* because your positivity was what I needed in the difficult moments. Last but not least, *Ms. Parisa Zehtabi*, one of the kindest people I ever met.

At King's I met many other people, BSc and MSc students, other PhD colleagues and Professors all worth to be mentioned, simply because they dedicated a bit of their time for me. I am really grateful to have been part of such a supportive, friendly and welcoming community. Here, I want to thank you all.

I would like to mention my sister *Giorgia Sempreboni* and my nephew *Cristian Gaiardoni*. During this whole journey, I missed many people. You are the ones I missed the most. Cristian, you are still young but I wish for you so many things that you don't even know, but most of all, I wish you to be curious about the life we have around and always hungry for knowledge. There are so many fascinating things in this world and not enough time. Cristian, "Stay Hungry, Stay Foolish".

I would like to thank my friends in Italy, *Alessandra Agostini* (the friend who told me about the roller coaster), *Alex Malfatti*, *Damiano Rossi*, *Fabio Bellini*, *Ilaria Iselle*, *Matteo Rossignoli*, *Nicola Residori*, *Roberto Chiodi* and *Ylenia Toffalori*. Even far away, you made me feel connected, in the old analogical fashioned way called friendship.

There is still one person I want to thank. Her name is *Stella*. You supported me since the first day we met, you believed in me when the only thing I was able to do

was to complain. Neither in Italian, nor in Greek nor in English would I have the right words to tell you how much I am grateful to have met someone like you. For this, and for all the other reasons that only we know, thanks.

Abstract

There is an increasing number of ICT systems (e.g. to communicate, do business, vote, control industrial processes or critical infrastructures, etc.) whose security depends intrinsically on human users. Concomitantly, there are many reported critical vulnerabilities that are due to users failing to follow security procedures or to behave as ICT scientists have decided is appropriate. A solution to this problem will only be found by addressing it radically differently, by treating it as a true socio-technical problem rather than just a technical one. We must understand how the technical components (e.g., software processes and digital communication protocols) and the social components (e.g., user interaction processes and user behaviour) of a system interoperate, and thus consider the system as a true socio-technical system, with people at its heart. This requires extending the technical analysis approaches with a mature understanding of human behaviour, as humans are complicated and nothing guarantees that, even if they learned how to operate a technology, either from a manual or through its use, they will comply with what they learned. Reasons include cognitive biases, fallacies, ignorance, distraction, laziness, curiosity of different uses, insufficient awareness of the security sensitivity of their behaviour, etc.

This thesis focuses on developing an innovative methodology to analyse the socio-technical security of ICT systems. To advance the state-of-the-art to the point where the wide spectrum of socio-technical security features of systems can be modelled formally and automatically analysed, this thesis aims to: *(i)* design a methodology to

tackle the socio-technical security of systems; *(ii)* define a formal modelling language expressive enough to cover the diverse security features of socio-technical systems; *(iii)* define libraries of prototypical socio-technical security properties, behavioural user models, socio-technical attack/threat models; *(iv)* implement a toolkit, an integrated front-end to holistically conduct formal security analysis of socio-technical systems; *(v)* demonstrate a proof-of-concept on a number of archetypal case studies.

Contents

List of Publications

During the compilation of this thesis, the following related articles have been published by the author:

1. Bella, G., Renaud, K., Sempreboni, D., and Viganò, L. (2019). An Investigation into the "beautification" of Security Ceremonies. In Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE - Volume 2: SECRYPT, pages 125–136. Scitepress Digital Library [25].

2. Sempreboni, D., Bella, G., Giustolisi, R., and Viganò, L. (2019). What Are the Threats? (Charting the Threat Models of Security Ceremonies). In Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE - Volume 2: SECRYPT, pages 161–172. Scitepress Digital Library [147].

3. Sempreboni, D. and Viganò, L. (2020). X-Men: A mutation-based approach for the formal analysis of security ceremonies. In Proceedings of the 5th IEEE European Symposium on Security and Privacy (EuroS&P). IEEE [151].

At the time of the submission of this thesis, the following manuscripts are being prepared for publication:

1. Bella, G., Renaud, K., Sempreboni, D., and Viganò, L. An Investigation into the "beautification" of Security Ceremonies (part 2). **(in preparation)**

2. Sempreboni, D. and and Viganò, L. A Socio-Technical Approach for Free Travel: How To Exploit Human Ticket Inspection In Coach Services. **(in preparation)**

These publications are not directly part of the research described in this thesis but are related to it and provide examples of the extent of my interests in cybersecurity.

1. Sempreboni, D. and Viganò, L. (2018). May I Mine Your Mind? In Proceedings of the 2nd Re-Coding Black Mirror workshop, Companion of The Web Conference (WWW), pages 1573–1576. ACM [148].

2. Viganò, L. and Sempreboni, D. (2018). Gnirut: The Trouble With Being Born Human In An Autonomous World. In Proceedings of the 2nd Re-Coding Black Mirror workshop, Companion of The Web Conference (WWW), pages 1567–1571. ACM [173].

3. Sempreboni, D. and Viganò, L. (2019). Privacy, Security and Trust in the Internet of Neurons. In Proceedings of the 3rd Re-Coding Black Mirror workshop, Companion of the Data Protection and Artificial Intelligence (CPDP) [149].

4. Sempreboni, D. and Viganò, L. (2019). Smart Humans... WannaDie? In Proceedings of the 3rd Re-Coding Black Mirror workshop, Companion of the Data Protection and Artificial Intelligence (CPDP) [150].

5. Viganò, L. and Sempreboni, D. (2018). Schrödinger's Man. In Proceedings of the 3rd Re-Coding Black Mirror workshop, Companion of the Data Protection and Artificial Intelligence (CPDP) [174].

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Overview

In the last two decades, technology has seen a drastic target change, from systems used by experts-only to systems accessible to a plethora of different types of users, thanks to the simplicity with which it has been possible to adapt them. A new idea of pervasive informatics [90] took ground establishing a new set of theories [95] that brought researchers to study pervasive spaces as complex domains using different theoretical approaches, i.e., socio-technical systems, computer-supported cooperative work, semiotics, considering the new era and the new technologies available.

The term *socio-technical system* (*STS* for short) was initially coined by Emery and Trist [59] to describe systems that involve a complex interaction between humans, machines and the environmental aspects of the work system.[1] Nowadays, the term socio-technical system is widely used to describe many complex systems, also in the cybersecurity domain. There are, in fact, an increasing number of cybersystems (e.g., to communicate, do business, vote, pay, control industrial processes or critical

[1] A *work system* is a system in which human participants and/or machines perform work (processes and activities) using information, technology, and other resources to produce products/services for internal or external customers.

infrastructures, etc.) whose security depends intrinsically not only on the security of the systems themselves but also on the human users and their sociality.

This intertwining of systems' security and humans has been fully captured by Kevin Mitnick, a renowned and reformed hacker, who, in his book [110], explains how an attacker can successfully deceive people to get access to their systems. Let me quote here from Mitnick's book:

> A company may have purchased the best security technologies that money can buy, trained their people so well that they lock up all their secrets before going home at night, and hired building guards from the best security firm in the business.
>
> That company is still totally vulnerable.
>
> Individuals may follow every best-security practice recommended by the experts, slavishly install every recommended security product, and be thoroughly vigilant about proper system configuration and applying security patches.
>
> Those individuals are still completely vulnerable.
>
> [...]
>
> Why? Because the human factor is truly security's weakest link.
>
> (*The Art of Deception*, Kevin D. Mitnick, p. 3. [110])

Since the publication of Mitnick's book, the security and privacy problem has gained a new and fresh dimension that requires extensive study. The many security failures due to users bear witness to the fact that humans cannot be designed out of the security loop: we must deal with security as a socio-technical problem rather than a mere technical one. Recent attention to the human nodes came from one of the NIST reports [82], where Jansen stressed that the protection of the client-side is often

overlooked, but has become more crucial in service computing. Reasons for this can be found in the intensive use of browsers, of social media, and of other public services or platforms, which make the consequences of social engineering and/or other types of attacks more critical.

The new socio-technical dimension has also to deal with understanding the *psychology* of users when interacting with digital resources: for instance, the user's perception of the security of a system might be utterly non-aligned with respect to the security that the system is able to guarantee. A wrong approach to these auxiliary aspects can make a system insecure.

Such aspects can be studied at least empirically. Schaik et al. [171] assessed a set of 16 security hazards (i.e., situations with the potential for harm to be done) on the Internet between UK and US students, measuring perceptions of risk and other risk dimensions, while Zhang et al. [180] assessed the security of password expiration and the replacement of a password with a new one. Recently, in addition to the empirical studies, a number of formal studies have been carried out [50, 133, 131, 20, 22, 84]. Bella et al. [22] applied formal methods to investigate the TLS certificate validation, a property whose enforcement may require getting the user involved. The study included considerations among all the most used browsers, corroborating the main concerns of Jansen [82] and giving support to Jagatic et al. [81], who asserted that technical security flaws are just one of the possible bases of social engineering attacks and that flaws that exploit the sociality have to be taken into account.

Many socio-technical attacks are also possible because security mechanisms are not designed for the users: many users view current security measures as unhelpful if not entirely irritating [76, 94]. This leads them often to bypass even the strongest security mechanism, sometimes in unexpectedly smart ways [47, 79, 65]: in fact, the recommended "secure" measures are not sustainable from the usability-centred and

psychology-centred points of view of users. Users prefer a more economical approach, which security designers usually ignore. Once we find the reasons why the systems are insecure, also including the human in our analysis, the application of these insights to secure system design can be effective [122].[2]

Considering socio-technical aspects under the psychological and usability lenses is just one side of the coin. On the other side, computer scientists have also been recently reflecting on the socio-technical aspects of cybersecurity. Even though several investigations have been carried out so far to study, mathematically and systematically, the nature of the socio-technical deficiencies of systems and protocols, of socio-technical attacks, and of the possible defences, much more needs to be done, considering also expanding the studies to involve several different fields. For instance, *Human-Computer Interaction (HCI)* studies have tried to apply rigorous formal techniques [32, 50, 134] for the analysis of human factors, with fewer still applying this to security [133], whereas *Deception* studies, as well as HCI, have contributed using formal approaches [40], dealing with automated auditing of trustworthy trade procedures for e-commerce.

Carrying out security analysis motivated Ellison in publishing a work on *security ceremonies* [58]. According to him, the term "ceremony" was first coined by Jesse Walker [96] to indicate those communications between human nodes and other nodes that usually happen not via network connections, but instead through face-to-face interactions, user interfaces, peripheral devices, or transfers of physical objects that carry data (e.g., USB memory sticks, SD cards and more recently smart cards, tokens, smartphone, etc.). Examples of ceremonies include two-factor authentication [16, 144], registration procedures [19], or protocols that users follow interacting with ATM machines [133], and many more due also to the pervasiveness of technological systems in our life. Martina et al. [102] have highlighted the importance of studying security

[2]This inspired me to carry out the study in Chapter 3.

ceremonies and the differences between a traditional protocol analysis methodology and ceremony analysis.

In fact, bringing the human nodes, so far never considered, into the so-called ceremony analysis, revealed flaws that a reductionist classical approach to technical security is not able to capture. Radke et al. [131] proved this assumption in their work, identifying a flaw in the Opera Mini browser ceremonies using the HTTPS protocol when used in a specific context, even though the protocol is secure in the traditional sense. However, the approach suggested by Radke et al. [131] was based on incomplete assumptions caused by the young age of ceremony analysis: since the topic was moving its first steps, researchers were still using the classical approach of threat models with the Dolev-Yao attacker [56] to model the malicious agents who act in the ceremonies.

An adequate threat model for security ceremonies, in fact, has to move in a non-technical dimension that brings us quite far from the use of the Dolev-Yao attacker. Creese et al. [48] recognised the Dolev-Yao attacker as inappropriate as a threat model for ceremony analysis, proving it by analysing protocols in a pervasive environment, which is what we here called a socio-technical protocol. Carlos et al. faced the problem of defining an adequate threat model for ceremonies [104]. Starting from the Dolev-Yao model, they derived a more realistic one, removing capabilities from it. They applied their approach to authentication protocols (e.g., WiFi printers, Bluetooth pairing). The work started by Carlos et al. [104] continued considering smart tokens and smart cards [101], and then was expanded via Carlos' PhD thesis [35], through the use of channels between human-to-human or human-to-machine.

However, the work towards threat models for socio-technical analysis is just at the beginning [64, 37]. In fact, even though the existing works defined threats that are reasonable, they generally failed to treat the threats *systematically* within the given ceremony, hence potentially missing relevant combinations of threats.[3]

[3]This inspired me to carry out the study in Chapter 4.

A further study by Bella and Coles-Kemp [20] went beyond the original ceremony between users and systems and produced what has been called *full security ceremony*, putting in place more aspects that have to be considered, such as interfaces between humans and machines (similarly to Carlos' notion of channels) but also the outermost layer whereby society influences behaviour (e.g., media, users' engagement with technology etc.). In order to study a full security ceremony, Bella et al. introduced what they called *concertina*, a multiple-layer framework that aims to provide the basis for a formal analysis. A multi-layered approach has become necessary due to the number of layers that must be traversed for a security property that the protocol enforces to reach the human.

The approach defined and adopted a user's persona model in formal analysis which can be influenced by cultural values, the belief systems and other factors. The defined personas take stock of the human persona introduced by Cooper [44] in the design process, but improving it making a persona able to express more than a single user's persona in front of the same technology at different times. Kumar et al. introduced the *rushing user*, who tends to skip anything, not directly connected to his main goal [93]. Both persona and the rushing user, however, seem to be the first steps for what could be a complete model of the human, capable of expanding his capabilities by increasing a potential library of behavioural patterns useful for the analysis of security ceremonies.[4]

This complete model needs to be applied in critical socio-technical domains where security and privacy analyses are decisive to unveil potential threats. Bella et al. identified and inspected some of these domains, i.e., file hosting, collaborative editing, electronic exam [21] and, similar to the electronic exam's domain, the electronic voting domain has also been taken into account [92] under the ceremony lens. Likewise in spirit to these domains, security issues are resumed by Blaze under the term "Human-Scale Security Problems" [30]. The study, in fact, aims to treats "human scale"

[4]This inspired me to carry out the study in Chapter 5

security problems and protocols as a central part of computer science, identifying and stimulating research on "non-traditional" protocols that might either have something to teach us or be susceptible to improvement via the techniques and tools of computer security. A number of human-scale security problems have been reported by Blaze [30], such as ordering wine (but in general food) in a restaurant, paying the check in a restaurant, drug testing in sports, voting in democratic elections but also railroad seat checks[5] highlighting how much human-scale systems are interesting and important and speculating on some future research directions.

1.2 Research Aims

The overall goal of this thesis is to develop a methodology and a formal and automated toolkit that encompass user studies and formal analysis in order to establish the security of ICT systems socio-technically, by targeting both the technical and the social components of the system. This methodology and analysis will also provide a stepping stone for run-time testing and behaviour change techniques.

Taking the socio-technical standpoint allows one to refuse the conclusion that "users are the weakest link in the security chain", which suggests, without providing insights or solutions, that users should not be involved in the practical task of establishing security. Instead, the socio-technical standpoint advocates that users are at the heart of the system: they must be part of the requirement elicitation process, of the security problem statement, of the solutions.

The overall goal has been achieved by considerably extending practices and tools for the formal analysis and assurance of security properties of ICT systems to include the human element together with the technical in a holistic, socio-technical methodology for security.

[5] This inspired me to carry out the study in Chapter 6.

More specifically, the main research goals of this thesis are (listed here not in order of importance but in their sequential order within the project):

1. **Beautification methodology:** Is it possible to identify socio-technical aspects dictated by a sense of beauty that drive the behavioural choices of users of security ceremonies?

2. **Charting methodology:** Is it possible to define a chart of the threat models for (the socio-technical components of) a security ceremony?

3. **Formal and automated approach:** Is it possible to use formal languages to model the wide spectrum of security features of STSs focusing in particular on security ceremonies (communication steps, behavioural user models, user-machine interactions, etc.) and provide tool support allowing an analyst to input a model of a ceremony and obtain in output either a set of attacks to the concrete ceremony or a proof of the ceremony's security (typically under a set of conditions).

4. **Socio-technical study**: Is it possible to enable reusability of models and results when moving from one case study to another?

The main research goals just listed here are tackled as in Table 1.1.

1.3 Thesis Outline

This thesis is structured as follows:

Chapter 2 introduces the basic concepts and definitions employed in this thesis. Starting from what a protocol is, a brief description of security protocols is given, followed by a summary of formal techniques and tools to analyse formal models of security protocols, concentrating the attention to model checking, one of the particularly

	Research goal	*Contributions*
(1)	**Beautification methodology**	• Definition of a beautification process for security ceremonies based on some guidelines. • Application of the process and test of its effectiveness on real case studies.
(2)	**Charting methodology**	• A systematic definition of an encompassing method to build the full threat model chart for security ceremonies. • Demonstration of the application of the charting technique for discovering new vulnerabilities that have not been considered so far and have arisen thanks to my charting method.
(3)	**Formal and automated approach**	• Definition of a formal approach that allows security analysts to model mistakes by human users of security ceremonies. • Development of a prototype tool for the automatic creation of mutated models based on human mistakes. • Proof-of-concept of my framework analysing on two real-life case studies.
(4)	**Socio-technical study**	• Extension of my framework to prove reusability to perform a socio-technical approach analysis of a human ticket inspection ceremony.

Table 1.1 Research aims linked to the actual work carried out.

successful formal protocol analysis techniques. Then, the concept of security ceremony analysis is explained along with the explanation of existing and promising automated tools that are suitable (and extendable) for the analysis of security ceremonies.

Chapter 3 investigates why many users view current security measures as unhelpful if not entirely irritating, suggesting four new general guidelines to improve these security measures. These guidelines have been extracted from the results of a user study I carried out. The approach, then, is applied to four security ceremonies formalised using Message Sequence Charts (MSC) in order to make them more sustainable from the usability-centred and psychology-centred points of view of users. To corroborate these results, I carried out also a second user study which provided interesting insights on future research directions.

Chapter 4 introduces a new systematic definition of an encompassing method to build the full chart of threat models for security ceremonies. The methodology defines a classification of the principals participating in security ceremonies and continues with a motivated labelling system for their actions and principals. Principals are then combined to derive a number of threat models that, together, form the full chart of threat models for security ceremonies. I demonstrated the application of the method on three existing ceremonies. In the end, I investigated the Danish Mobilpendlerkort ceremony, the inspection ceremony for the mobile transport ticket in Denmark, finding a threat model that has not been considered so far and that has arisen here thanks to my charting method. To demonstrate the relevance of the chart, I modelled and analysed this threat model using the formal and automated tool Tamarin.

Chapter 5 introduces a novel mutation-based approach for the formal analysis of security ceremonies that focuses on the vulnerabilities that result from the mistakes that human users might make, such as skipping one or more of the actions, replacing a message with another one, adding an action of a ceremony, and their combinations. Using an extension of a formal language based on multiset rewriting rules, I define a new language expressive enough to model security features of ceremonies. The resulting

formal language enables reusability as I have shown modelling a number of different ceremonies (see also Chapter 6). The novel mutation-based approach I define allows *matching mutations* to adjust non-human roles so that they can be executed together with a mutated human role, and *propagation rules* to create an executable trace that can be analysed in search for attacks. In order to automatise the analysis of security ceremonies, I have developed a prototype tool called X-Men, which creates mutated models that can then be input to Tamarin, and applied it to two relevant ceremonies, one of which one had never been considered before.

Chapter 6 describes an original socio-technical study on coach services operating in different countries by targeting both the technical and the social components of the system. The study consists of two phases: observation and analysis. The observation concerns the interaction between customers (who are using the services to travel from/to airports) and drivers of coaches (who are usually inspecting the tickets). After having introduced a generic coach service ecosystem, a security analysis has been carried out, formally and semi-formally, on the ticketing inspection ceremony, showing that the customer can attack the socio-technical system in order to travel for free. I verified my assumptions and considerations using and extending my novel methodology described in Chapter 5, proving once again that my methodology enables reusability and it is successful for discovering new threats in ceremonies.

Chapter 7 provides conclusions and a discussion on potential future work.

Chapter 2

Background on Security Ceremony Analysis

The main topic of this thesis is the formal and automated security analysis of Socio-Technical Systems (STSs). This chapter reviews essential concepts, methods and work related to this topic and paves the way for the proposed methodologies and results produced in the following chapters. More specifically, this chapter includes an introduction to protocols, security protocol analysis, security ceremony analysis, formal techniques for the analysis of security protocols, state-of-the-art of techniques for the analysis of security ceremonies and other techniques aimed to analyse the security of protocols involving humans. This chapter works as a background chapter, however, every single approach and solution described in this thesis have a dedicated detailed discussion of related works for that specific approach or solution, in addition to a detailed comparison of relative advantages in each chapter where they are explained.

2.1 From Protocols to Security Protocols Analysis

A protocol is defined simply as a set of rules or instructions that determine how "actors" act or interact in a given situation: a recipe for a cake, the procedure for boarding, the steps for buying a movie ticket are all possible examples of protocols.

Any protocol that is designed to satisfy some kind of security property can be called a *security protocol*. A security protocol, indeed, describes how agents exchange messages, built using cryptographic primitives, in order to obtain security guarantees such as confidentiality (to ensure that particular pieces of information are kept secret from certain parties) or authentication (to ensure that an agent A is talking to the agent the agent A intends to talk to). Security protocol specifications are parametric and prescribe a general recipe for communication that can be used by different agents playing in the protocol roles (sender, receiver, server, etc.).

The most (in)famous security protocol is, perhaps, the *Needham-Schroeder Public Key (NSPK) protocol* [117], which is shown in Figure 2.1.

$$
\begin{aligned}
A \to B &\ : \ \{N_A, A\}_{pkB} \\
B \to A &\ : \ \{N_A, N_B\}_{pkA} \\
A \to B &\ : \ \{N_B\}_{pkB}
\end{aligned}
$$

Figure 2.1 Needham-Schroeder Public Key (NSPK) protocol

The NSPK protocol is a security protocol that was proposed for the mutual authentication of a pair of agents A and B in a distributed computer system. For instance, agents want to be assured of each other's identity before they exchange security-critical data. Thus, an attacker should not be allowed to impersonate another

agent. For this purpose, initiator and responder of a communication need to mutually authenticate each other.

NSPK uses public key cryptography, i.e., each agent possesses a public key which can be accessed by all agents and a secret key, which is the inverse of the public key. Moreover, the protocol makes use of *nonces*, which are freshly generated, random numbers to be used in a single run of the protocol. It is assumed that these numbers are generated in such a way that they cannot be guessed by other agents.

In this protocol A plays the role of the initiator and B reacts in the responder role. A encrypts his nonce N_A with the public key of B and sends it along with his identity to B. Now A is in a waiting status. B decrypts the message, creates a new nonce and sends back to A the concatenation of both nonces encrypted with the public key of A. After this, B is waiting for the answer from A. A can decrypt the message sent by B. In case the contents of the message is the concatenation of his own nonce N_A with something else, A can be assured that the message was sent by B, because by the perfect cryptography assumption B is the only agent who can decrypt the message sent by A containing the nonce N_A. Thus, A establishes a communication with B. Moreover, to acknowledge to B the receipt of B's nonce, A separates it from the message and sends it back. If B receives this message from A, he can be assured of being talking to A and, therefore, of having established a communication with A.

Several techniques and tools have been developed to detect security flaws in security protocols, but formal methods [176] have proved to be the best approach. In fact, there exist a number of flawed protocols that received an extensive manual analysis without discovering any flaw (or discovering just a subset of the flaws). The NSPK is a well-known example. In fact, Lowe discovered that the protocol was vulnerable to a *Man in the Middle (MITM) Attack*, wherein an attacker i, who is responding to a protocol run initiated by A, can falsely authenticate itself to an agent B as A, by

replaying A's message to B. Thus, B is fooled to believe that a session is established between A and B [98]. Lowe's attack is shown in Figure 2.2, where $i(A)$ indicates that i is pretending to be A.

$$\begin{aligned} A \rightarrow i &: \{N_A, A\}_{pki} \\ i(A) \rightarrow B &: \{N_A, A\}_{pkB} \\ B \rightarrow i(A) &: \{N_A, N_B\}_{pkA} \\ i \rightarrow A &: \{N_A, N_B\}_{pkA} \\ A \rightarrow i &: \{N_B\}_{pki} \\ i(A) \rightarrow B &: \{N_B\}_{pkB} \end{aligned}$$

Figure 2.2 Lowe's Attack on Needham-Schroeder Public Key (NSPK)

The difficulty of designing and analyzing security protocols is well-known. This difficulty stems from a number of considerations:

- The protocols are (often) complex environments. In order to analyse them properly, it is necessary to describe them accurately modelling the environment which comprises the capabilities of honest and dishonest agents.
- A dishonest agent, often called "intruder" or "attacker" deliberately tries to undermine the protocol. Capturing the capabilities of the attacker is difficult, but we are able to to make good models of the attacker that can be progressively enhanced as new styles of attack come to light.[1]
- The properties the security protocols are supposed to ensure harbour a number of subtleties and come in various guises.

[1]In Chapter 5 I introduce a new way to see the attacker in security protocols.

Further considerations contemplate modelling the different channels between agents, algebraic properties of the cryptographic operators and more.

2.1.1 Modelling

Modelling the honest agents The standard way to formally model a security protocol is to formalize how agents execute the *roles* of the protocol to attempt to achieve one or more security goals in the presence of an attacker. Roles are sequences of events (sending or receiving messages, generating fresh values, etc.), which are usually represented graphically by a structure generated by causal interaction such as *strands* [166, 63] or the vertical lines in MSCs and *Alice&Bob notation* [4], or less graphically by a process in a process algebra such as in the *applied pi calculus* [1] or by a so-called *role script*, basically the projection to an individual role of an extended Alice&Bob specification, and corresponds to a strand or an applied pi calculus process.

Dishonest agents Dishonest agent capabilities are based on the *Dolev-Yao attacker model* [56]. Even though the presence of many dishonest agents that acts in different points of the environment can be considered, it has been proved that it is sufficient to consider, in a security protocol, the presence of a single dishonest agent modelled using the Dolev-Yao attacker [13]. The environment where the agents interact can be seen as a network that allows the agent to communicate with each other by sending messages over communication channels assuming that the network is under the control of an active attacker (the so-called Dolev-Yao attacker) who has control of the entire network and can perform some actions on the messages that transit on it. In particular, the Dolev-Yao attacker can read, modify and destroy any message, and create new messages as long as he does not break cryptography (following the perfect cryptography assumption). Approaches that go beyond the Dolev-Yao attacker exist: some of these approaches introduce restrictions such as restrictions on the communication channels

that the attacker can control, some others introduce extensions such as extensions on allowing the attacker to attack the cryptography[2]).

Security properties Security properties are often outlined using their high-level description and this supposedly widely understood description is given different interpretations, sometimes within a single document or design. It is for this reason that it is so essential to give precise, formal meanings. It is not sufficient, for example, to assert that a particular protocol is "secure", or worse "correct". A protocol can only be claimed to be secure with respect to a given, precisely defined property and even then only against certain classes of threats and subject to various assumptions. Here are some of the common security properties:

- **Confidentiality**. This term covers two related concepts.
  - *Data confidentiality*, which assures that private or confidential information is not made available or disclosed to unauthorized individuals. Specific to security protocols, it means that the attacker is not able to derive the plaintext of messages passing between the honest nodes, if the message is protected using cryptography.
  - *Privacy*, which assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
- **Integrity**. This term refers to *data integrity*, which assures that information and programs are changed only in a specific and authorised manner. Specific to security protocols, it means that data cannot be corrupted, or at least any such corruption will always be detected.

[2]It is not in the scope of this thesis to deal with cryptographic analysis, which is debated in e.g., [34, 29, 27, 12, 57, 42]

- **Availability**. This assures that data/services can be accessed when desired.
- **Authentication (entity)**. This property assures that an agent can be confident that the claimed identity of an agent with whom he is interacting is correct.
- **Authentication (origin)**. This property assures that an agent can be sure that a message that purports to be from a certain agent was indeed originated by that agent. This is typically related to assuring that the message has not been tampered with.
- **Non-repudiation**. The property assures that actions can be traced to responsible agents, who cannot deny their actions.
- **Anonymity**. The property describes situations where the acting person's name is unknown. This is commonly achieved creating a set, called "*anonymity set*", where the "anonymity" of an element of this set refers to the property of that element of not being identifiable within this set. If it is not identifiable, then the element is said to be "anonymous".

Channels Channels are an abstraction of the many concrete techniques to enforce particular properties of message transmissions such as encryption [112]. The Dolev-Yao represents an active attacker that has complete control of the communication network. However, we can limit the attacker's control over the protocol agents' communication channels by specifying channel rules, which model channels with intrinsic security properties. Such security properties can define:

- an *authentic channel*, where an agent B can rely on the fact that the agent A has sent the message M and meant it for B.
- a *confidential channel*, where an agent A can rely on the fact that only B can receive the message A is sending.

- a *secure channel*, where the channel is both authentic and confidential.

2.1.2 Automated Tools for the Analysis of Security Protocols

Many techniques and tools have been developed to analyse formal models, e.g., *inductive reasoning* [124], *theorem proving* [97], *generic verification systems* [78, 167]. However, one of the formal techniques particularly successful in the analysis of security protocols has been found to be *model checking* [10] along with similar techniques based on reachability analysis.

Given a formal representation of a system (also called system model) along with a property of interest that should be analysed on the system, a model checker automatically checks whether the property holds on the system.

The properties of interest describe what the system model should do or what the system should not do and depend on the requirements on the system itself, for example: does Alice authenticate Bob and vice versa, agreeing on two secret nonces?

Once the system model and the properties that should hold in the system are formalised, they are then given to a model checker, which explores the possible states of the system in a systematic way. Such systematic exploration aims to find a state or a trace that violates the property defined on the model. If so, the model checker provides a counterexample that indicates how the model can be attacked. A counterexample describes an execution path that starts in the initial state and leads to a final state where a property is violated.

Dedicated tools (based on model checking and automated reasoning) have been developed starting from the Dolev-Yao paradigm [109, 155, 6, 5, 61, 28, 49, 108]. In the following I provide a brief description only the most important and relevant ones for this thesis:

- **The AVISPA tool** [6] is a push-button tool for the Automated Validation of Internet Security Protocols and Applications. It provides it own formal language for specifying protocols and their security properties. It integrates different back-ends that implement a variety of automatic protocol analysis techniques.

- **The AVANTSSAR platform** [5] is the successor of the AVISPA tool, and is an integrated toolset for the formal specification and automated validation of trust and security of SOAs and, in general, of applications in the IoS and their security protocols. The platform comprises three back-ends: CL-AtSe [170], OFMC [14] and SATMC [8], which operate on the same input specification providing complementary automated reasoning techniques (including service orchestration and compositional reasoning, model checking, and abstraction-based validation);

- **Maude-NPA** [61] is a protocol analysis tool that takes into account algebraic properties of crypto systems such as cancellation of encryption and decryption, Abelian groups (including exclusive-or), exponentiation, and homomorphic encryption. Maude-NPA performs backwards search from a final state to determine whether or not a state is reachable;

- **ProVerif** [28] is an automatic cryptographic protocol verifier based on a representation of the protocol by Horn clauses. The ProVerif verifier can handle many different cryptographic primitives, including shared-key and public-key cryptography (encryption and signatures), hash functions, and Diffie-Hellman key agreements, specified both as rewrite rules or as equations. It can also handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. The ProVerif tool can prove properties of secrecy,

strong secrecy, authentication and equivalences between processes (that differ only by terms);

- **Scyther** [49] is a push-button tool for the verification, the falsification and the analysis of security protocols based on a pattern refinement algorithm, providing concise representations of (infinite) sets of traces introducing novel features such as the possibility of unbounded verification with guaranteed termination, analysis of infinite sets of traces in terms of patterns, and support for multi-protocol analysis;

- The **Tamarin prover** [108] provides automated, unbounded, symbolic analysis of security protocols. The Tamarin prover employs the backwards search used by the Scyther tool [49] to enable protocol specification by multiset rewriting rules, property specification in a guarded fragment of first-order logic, which allows quantification over messages and timepoints, and reasoning modulo equational theories. Due to its importance in this thesis, Tamarin is described in more detail in Section 2.3; moreover, my approach to extend the prover with the capabilities to analyse security ceremonies is described in Chapter 5.

These tools (and other ones) have been successful at uncovering a large number of vulnerabilities. However, they do not take into account the wide range of side-channel attacks, social engineering, human behavioural and cognitive aspects in their relation with "machine" security, and interfaces to other protocols and the environment, which occur in the real world.

2.2 Security Ceremony Analysis

A *security ceremony* expands a security protocol with everything that is considered out-of-band to it, including, in particular, the mistakes that human users might make when participating actively in the security ceremony.

The term *ceremony* was coined by Jesse Walker [96] to describe the interaction between a user and computing devices. In cybersecurity, the term "*ceremony*" has been used by Ellison [58] as an extension of the concept of security protocol, with human nodes alongside computer nodes and with communication links that include UI, human-to-human communication and transfers of physical objects that carry data. In particular, Ellison remarked that "what is out-of-band to a protocol is in-band to a ceremony, and therefore subject to design and analysis using variants of the same mature techniques used for the design and analysis of protocols". An example of security ceremony analysed by Ellison considers the process of getting an e-mail certificate and delivering the key by which that certificate is validated. The ceremony is shown in Figure 2.3 as *Message Sequence Chart (MSC)*. In the ceremony, there are:

- Alice, the sender with computer AC,
- Bob, the receiver with computer BC; and
- Carol, the Certification Authority with computer CC.

The sub-ceremony (a) in Figure 2.3 shows the process by which Bob gets a certificate. This must start with a human-to-human interaction by which Bob convinces Carol to issue him a certificate. Bob then interacts with his computer, BC, to cause it to generate a certificate request to Carol's computer, CC. Then, Carol instructs her computer, CC (the CA), to issue the requested certificate and send that certificate to Bob's computer (e.g., via his e-mail address). To conclude, Bob's computer then uploads the newly received certificate to a directory. The sub-ceremony (b) in Figure 2.3

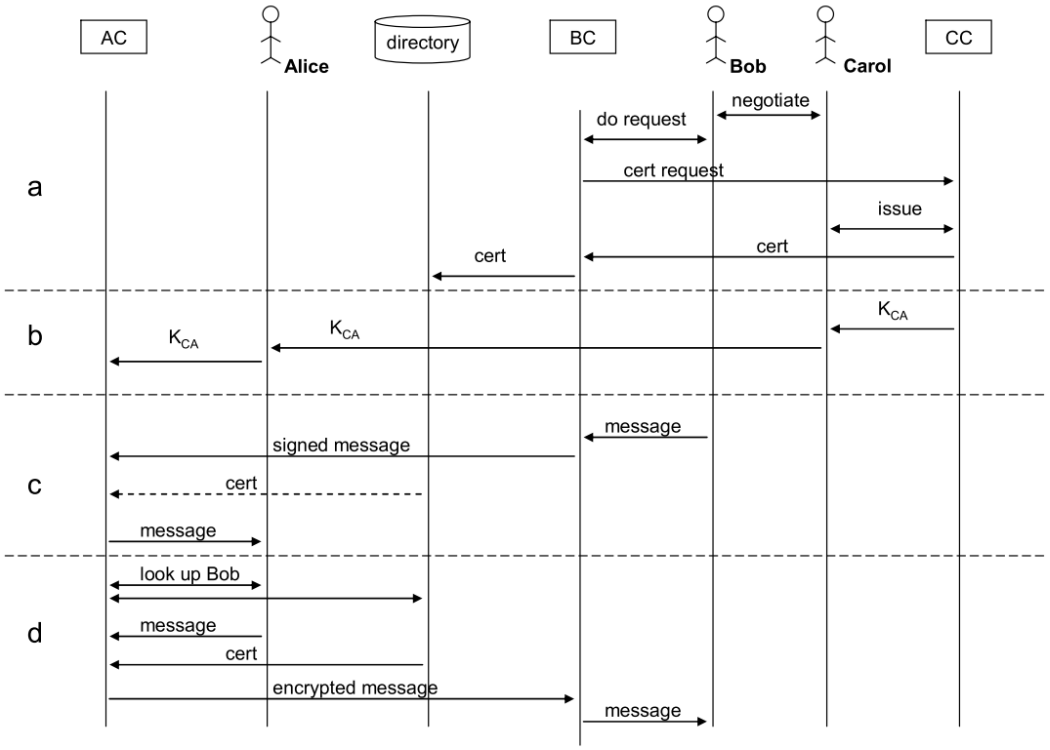


Figure 2.3 The Cryptographic E-mail ceremony

can occur before or after the issuance of Bob's certificate. This is the sub-ceremony by which Alice receives the CA's key, K_{CA}, from Carol and depends on the security of human-to-human channels. The sub-ceremony (c) in Figure 2.3 shows the normal signed e-mail ceremony. A message is composed by Bob, which is also digitally signed and sent by its computer using Bob's private key K_B^{-1}. In this phase, Bob's computer could also transmit:

(i) Bob's S/MIME signature certificate or PGP signed key (binding Bob's common name and e-mail address to his public key, K_B),

(ii) an S/MIME encryption certificate (binding Bob's common name and e-mail address to his encryption public key, K_{BE}),

(iii) nothing, and Alice's computer might then fetch the certificate it needs from a directory.

To conclude, in the sub-ceremony (d) in Figure 2.3 Alice transmits her response after having looked up Bob's e-mail address in the directory.

The reason why Ellison stressed that we need to investigate security ceremonies lies in the last message in the sub-ceremony (c), from AC to Alice. This message, according to the procedure, is delivered as a body of text with header information based on the results of the signature validation. The mail client, based on the design and its implementation, could show:

- the common name from the certificate,
- the e-mail address from the certificate,
- the common name provided in the mail header;
- the e-mail address provided in the mail header.

Once Alice obtains the message, Alice needs to evaluate the message from Bob. Her process will involve comparing what is displayed to her to what she has in her memory. She could compare:

- the "from:" address,
- the picture (if present),
- the "signed by:" address,
- the text in the message itself (phraseology, word choice,...),
- etc.

What could possibly happen if Alice evaluates the message sent from a malicious agent Charlie, claiming to be Bob? If Alice were a computer (a robot, for instance) the fact that, maybe, the "signed by:" address differed in one character from the correct

one would make the header incorrect and she/it would detect fraud. But what if Alice were a real human?

In this case, some factors (e.g., sociality, personal circumstances, technological and communications capabilities, etc.) have to be considered, as also highlighted, using other examples, by Bella et all. [20]. This is exactly where security ceremony analysis can fit, finding fertile ground for research. Let's also point out that a formal definition of security ceremony does not exist. Inspired by Ellison [58], in this thesis I focus, in particular, on security ceremonies that consist of security protocols where some of the agents are interpreted by humans (e.g., the humans Alice, Bob, Carol who were operating with the computers AC, BC and CC respectively).

Similar to security protocol analysis, I have to deal with three main problems:

- Define a formal language expressive enough to model the wide spectrum of security features of security ceremonies (communication steps, behavioural user models, user-machine interactions, etc.) but also enable reusability of models and results when moving from one case study to another.

- Deal with the concept of dishonest agents while I am investigating security ceremonies.

- Provide tool support, creating an integrated front-end to holistically conduct formal security analysis of security ceremonies, in order to input a model of a security ceremony and obtain in output either a set of attacks to the concrete security ceremony or a proof of the security ceremony's security (typically under a set of conditions).

In this thesis, I study security ceremonies tackling *(i) the modelling problem for security ceremonies, (ii) the threat models for security ceremonies, (iii) the tool support for security ceremonies.*

2.2.1 Formal Languages and Automated Tools for the Analysis of Security Ceremonies

To the best of my knowledge, there does not exist any tool for the full-fledged formal analysis of security ceremonies. However, the recent awareness that security protocols need better attention to their "human element" spawned several works, both on the formal and on the practical level (e.g., adapting and using existing tools).

The semi-formal and formal analysis of socio-technical systems is not new. In order to understand why the presented approach has been chosen, a comparative discussion of related works on socio-technical systems is needed. As my approach takes inspiration from the mutation-based testing technique, generating mutations of the security ceremony to characterise the human user and deal with critical security aspects on security ceremonies, I also include a brief description of existing works on mutation-based testing, discussing the similarities and differences with my approach.

A large body of work on researching methodologies to model socio-technical systems (STSs) spans reaching the use of *Business Process Model And Notation* (BPMN) [111, 120, 177, 85]. The reason for this has to be found in the term "socio-technical system" and in the purpose for which BPMN was born. On one hand, we have the term *socio-technical system* that, as I mentioned in Chapter 1, was initially coined to describe systems that involve a complex interaction between humans, machines and the environmental aspects of the work system. On the other hand, we have BPMN, whose primary goal is to provide a notation that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes to the technical developers responsible for implementing the technology that will perform those processes and, finally, to the business people who will manage and monitor those processes [175].

As said in [137], an STS can be conceptualised as a complex organisation where interactions among its operative units are represented as business processes (as represented in a BPMN) and security policies are used to regulate or prevent undesired behaviours.

The modelling language *STS-ml*, proposed in the context of the European Project *Aniketos* [120], includes high-level organisational concepts such as actor, goal, delegation, etc. Security requirements in STS-ml models are mapped to social commitments [52]. Expressing security needs via commitments allows the modeller to deal with some common security properties such as *non-repudiation*, *integrity*, *data confidentiality* expressed as "non-disclosure". However, other security properties are missing (e.g., anonymity, confidentiality as privacy, availability).

Although BPMN is the de-facto standard for representing the interaction among components of complex systems, it does not allow for representing security restrictions. Extensions of BPMN have been proposed to remedy this problem, introducing what has been called "SecBPMN" [138, 139]. However, these approaches do not consider how agents communicate, representing the communications as standard BPMN relations (e.g., flow, message flow). In fact, BPMN approaches lack the ability to formally model, in a simple way, channels between agents and the applications of cryptographic operators in messages sent on these channels.

Moreover, and perhaps even more importantly, agents in protocols/ceremonies are characterised by what is usually called "knowledge": each agent's state is represented by its initial knowledge which changes during the protocol/ceremony execution. Given that entities are represented by simple tasks in BPMN approaches, representing dynamic knowledge requires an extension of this notion of task. Even though "SecBPMN" and "STS-ml" represent good approaches to study STSs from a wider perspective, taking into account how independent components socially interact with one another, their

granularity is insufficient to deal with security protocols and ceremonies as they are far more complex (cf. Section 2.1 and Section 2.2).

Other formal approaches that consider STSs exist and rely on tools such as Isabelle [20, 129], PVS [105] or PIE [106], but also other popular tools such as ProVerif [28] and the AVANTSSAR platform [5] could be extended to cope with socio-technical protocols. Generally, it can be expected that whenever the interaction between agents can be specified as a distributed protocol, then traditional tools for security protocol analysis could help out. Isabelle has been recently used to support a work that combined formal modelling and analysis of infrastructures of organizations with sociological explanations to provide a framework for insider threat analysis [87, 86]. However, as explained, the framework can fruitfully be applied to the characterisation of insider threats (e.g., fraud, theft of Intellectual Property, sabotage), but left unanswered the important question of how to deal with external threats, tackled in this thesis in Chapter 4 where I discuss the importance of considering threat models with an external attacker. Bella and Coles-Kemp [20] defined a layered model of socio-technical protocols between a user persona and a computer interface, and used Isabelle to analyse the Layer III of the *concertina model* of security ceremonies formally.

So, even though Isabelle is promising to deal with security ceremonies and an extension seems feasible, it is not powerful nor flexible as Tamarin. This is the reason why I decided to use Tamarin at first.

More works that consider STSs and threats in STSs exist. At the formal level there is the work by Martimiano and Martina [100], who showed how a popular security ceremony could be made fail-safe assuming a weaker threat model than normally considered in formal analysis and compensating for that with usability. Basin et al. [16] provided a formal model for reasoning about some errors that humans involved in security protocols may make. They specified rules formalising different types of humans

(untrained, infallible or fallible humans and some combinations of these characteristics), and focused on errors disclosing information to the attacker. They successfully applied their model to analyse some authentication protocols using the Tamarin prover. Similar to [16], Curzon et al. [50] proposed a formal human model that includes a specific attacker able to exploit the errors against the human user. The errors considered are those caused by the humans' interpretation of the system and by the design of the interfaces, but not those entailed by human choices or mistakes. Moreover, they do not consider communication channels. As far as I know, the approach proposed by Curzon et al. [50] has never been implemented in any automated tool.

As I will illustrate in more detail in Chapter 4, these works cover only a small portion of the landscape of possible threat models, so a transformative approach is called for.

At the practical level there is the work by Hall [74], who provided real-world summaries of weak passwords in use, with 2018's weakest one still being "123456", and the work of Bella et al. [24], who focussed on protocols for secure exams and provided a novel protocol that they have formally analysed using the tool ProVerif. The work carried out by Bella et al., however, focused on making secure a protocol on specific seven authentication, five privacy, two verifiability, and one accountability requirements. The attacker model used is the Dolev-Yao attacker in addition to the following specific threats: corruption of candidates, corruption of authorities (i.e., administrator, invigilator and examiner). Each of these agents is not modelled per se, as I am doing instead in Chapter 5 modelling the human agent who is travelling using the Oyster card. The analysis of possible threats caused by these agents is not automated and is based on guessing, differently from what I have achieved with my framework, since it allows me to discover threats caused by implemented behavioural patterns on human mistakes.

Other relevant related works targeting threats and humans exist. Some stress the threats deriving from humans, for example humans repeating a previous sequence of actions without considering whether it would be currently appropriate [88], or humans making errors during text entry [156]. In Chapter 5, I introduce a framework to reason on human errors, but considering whether they can fit with the actual actions that a human is carrying out during the ceremony. Others discuss the human behaviour that may be maliciously induced by a third party, for example by deception [110] or by following precise coercion principles [158]. My work does not consider deception per se, however in Chapter 6 I demonstrate how it would be possible to exploit human errors to gain some advantages, such as travelling for free. Some others consider threats in e-mail phishing [119], out of scope for this thesis as it would be a Human–computer interaction issue, or focussed on modelling insider threats using a structured model that emphasises individual and organizational socio-technical factors [71] through ontologies. Another study performs security threat modelling for a global software supply chain [136], proposing a socio-technical framework for studying the software supply chain security problem from a systemic viewpoint, however the framework seems less powerful than the concertina framework introduced by Bella and Coles-Kemp [20] not carrying out any formal analysis but corroborating their insights using an ontological analysis. Overall, ontologies are definitely insufficient to be used for formal security analysis and to model humans and attacker capabilities.

As mentioned above, my approach has been inspired by the idea behind the mutation-based testing technique, which is a well-known and used technique in software testing [55, 39, 118, 153, 83] that is finding fertile ground even in security [154, 152, 33, 51]. Mutation testing is a fault-injection testing technique that improves the quality of test cases using versions of a program in which artificial changes are introduced to the target program. These versions are called mutants.

My approach focuses on design-time analysis, i.e., analysis of specifications of security ceremonies, rather than run-time testing. However, in the spirit of model-based testing, the results of X-Men could be used for testing by concretising the attack traces found into test cases to be applied for testing the code of the security ceremonies [127, 172]. Generally, mutation-based testing techniques apply mutants to verify the correctness of the implementation. Approaches that use mutation-based testing techniques at specification level exist [62, 115]. In fact, recently, it has been proposed that the concept be applied to specification-based (black box) testing [115]. The approach, however, generates test cases by systematically replacing data items relevant to a particular part of a specification with a data item relevant to another, looking for code or programming errors. Other approaches that consider the application of mutation testing for validating specifications are used with statecharts [62] and Estelle [54]. Differently from these works, I am generating mutations, based on behavioural patterns, targeting parts of the specification. The behavioural patterns represent my mutants, which are not related with the specification itself but more with the high-level interpretation of human mistakes.

Before going into further details, an exhaustive description of the Tamarin prover is needed in order to understand my approach. This description is in Section 2.3. I made extensive use of this tool: in Chapter 4 to prove security properties of ceremonies without any modifications, in Chapter 5 and Chapter 6 as a starting point of my framework which considers the extension of the Tamarin prover capabilities to analyse human errors in security ceremonies.

2.3 The Tamarin Prover

The Tamarin prover [108, 161] is a tool for the symbolic modelling and analysis of security protocols. It takes as input a security protocol model, specifying the actions

taken by agents running the protocol in different roles (e.g., the protocol initiator, the responder, and the trusted key server), a specification of the attacker, and a specification of the protocol's desired properties. If Tamarin is not able to find an attack, then it provides a proof that, even when arbitrarily many instances of the protocol's roles are interleaved in parallel, together with the actions of the attacker, the protocol fulfils its specified properties.

Tamarin's workflow for proving the validity of security properties is depicted in Figure 2.4 (this picture is based on the one presents in Schmidt's thesis [142]). As shown in Figure 2.4, Tamarin:

- Considers protocols and their attacker specified using an expressive language based on multiset rewriting rules. These rules define a labelled transition system in which states consist of a symbolic representation of the attacker's knowledge, the messages on the network, information about freshly generated values, and the protocol's state. The attacker and the protocol interact by updating network messages and generating new messages.

- Supports the equational specification of some cryptographic operators, such as Diffie-Hellman exponentiation, asymmetric-encryption and bilinear pairings.

- Works also using a sequence of restrictions specified as guarded trace properties.

- Deals with security properties modelled as trace properties, checked against the traces of the transition system, or in terms of the observational equivalence of two transition systems.

In order to explain the syntax, I am going to refer to the Needham-Schroeder Public Key (NSPK) protocol in Figure 2.1, with minor deviation due to the tool's syntax. Let's see how it is possible to model NSPK using Tamarin.

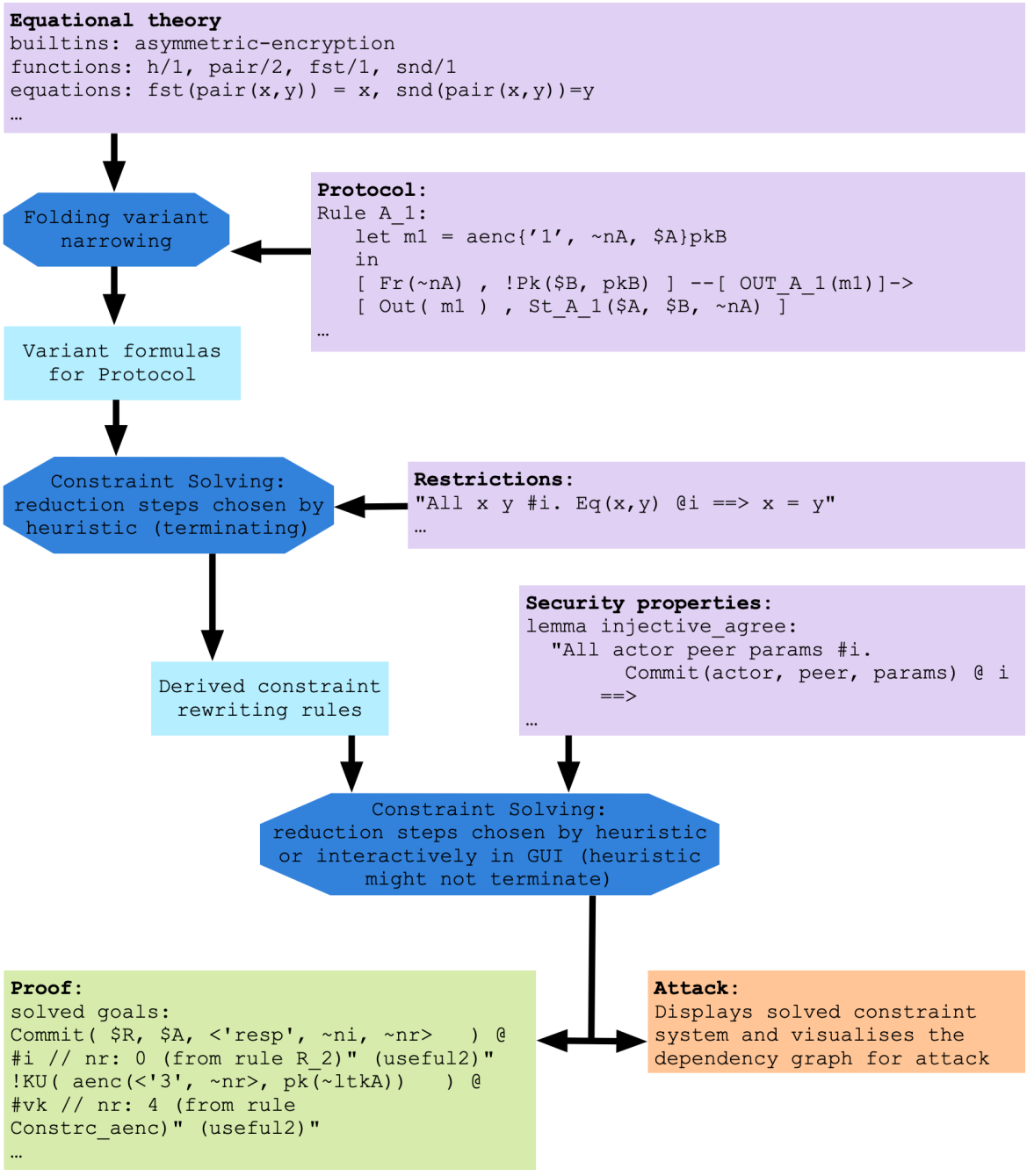


Figure 2.4 Tamarin workflow for proving validity of security properties

Equational Theory. The modeller specifies the built-in function symbols used in the protocol from a set of functions available. The latest version of Tamarin (version 1.4.1) provides the following function symbols:

- **diffie-hellman**: this theory models Diffie-Hellman groups. It defines the function symbols `inv/1`, `1/0`, and the symbols `^` and `*`. I use `g ^ a` to denote exponentiation in the group and `*`, `inv` and `1` to model the (multiplicative) abelian group of exponents (the integers modulo the group order).

- **hashing**: this theory models a hash function. It defines the function symbol h/1 and no equations.

- **symmetric-encryption**: this theory models a symmetric encryption scheme. It defines the function symbols `senc/2` and `sdec/2`, which are related by the equation `sdec(senc(m,k),k) = m`.

- **asymmetric-encryption**: this theory models a public key encryption scheme. It defines the function symbols `aenc/2`, `adec/2`, and `pk/1`, which are related by the equation `adec(aenc(m, pk(sk)), sk) = m`.

- **signing**: This theory models a signature scheme. It defines the function symbols `sign/2`, `verify/3`, `pk/1`, and `true`, which are related by the equation `verify(sign(m,sk),m,pk(sk)) = true`.

- **bilinear-pairing**: this theory models bilinear groups. It extends the diffie-hellman theory with the function symbols `pmult/2` and `em/2`. Here, `pmult(x,p)` denotes the multiplication of the point `p` by the scalar `x` and `em(p,q)` denotes the application of the bilinear map to the points `p` and `q`. Additional equations are available.

- **xor**: this theory models the exclusive-or operation. It adds the function symbols `/2` (also written as `XOR/2`) and `zero/0` and it is associative and commutative.

- **multiset**: this theory introduces the associative-commutative operator `+` which is usually used to model multisets.

- **revealing-signing**: this theory models a message-revealing signature scheme. It defines the function symbols `revealSign/2`, `revealVerify/3`, `getMessage/1`, `pk/1`, and `true`, which are related by the equations `revealVerify(revealSign(m,sk),m,pk(sk)) = true` and `getMessage(revealSign(m,sk)) = m`.

In addition, it is possible to define here new function symbols with their arity. To define new function symbols `f1, ..., fn` with arity `a1,...,an` the syntax is:

```
functions:  f1/a1, ..., fn/an
```

In the NSPK example, I use Tamarin's built-in functions for asymmetric-encryption, declared as:

```
builtins:  asymmetric-encryption
```

Model specification. The protocol is defined specifying *multiset rewriting rules*. Multiset rewriting is a formalism that is commonly used to model concurrent systems since it naturally supports independent transitions.

A *multiset rewriting system* defines a transition system, where, in Tamarin, the transitions will be labelled. The system's state is a multiset (bag) of facts.

A *rewrite rule* in Tamarin has a name and three parts, each of which is a sequence of facts: one for the rule's left-hand side *(prem)*, one labelling the transition, which in Tamarin are called 'action facts' *(a)*, and one for the rule's right-hand side *(conc)*.

$$prem \xrightarrow{a} conc$$

A rewrite rule in Tamarin has the structure as:

```
rule rule_name:
                    [] --[]-> []
```

Facts are of the form `F(t1,...,tn)` for a fact symbol `F`, terms `ti` and arity `n`. Tamarin has three types of special facts built in. Two are used to model interaction with the untrusted network and the other one to model the generation of unique fresh (random) values.

The `In(...)` fact is used to model an agent receiving a message from the untrusted network that is controlled by a Dolev-Yao attacker, and can only occur on the left-hand side of a rewrite rule. `Out(...)` is used to model an agent sending a message to the untrusted network that is controlled by a Dolev-Yao attacker, and can only occur on the right-hand side of a rewrite rule.

`Fr(...)` must be used when generating fresh (random) values, and can only occur on the left-hand side of a rewrite rule, where its argument is the fresh term.

In Tamarin, the sort of a variable is expressed using prefixes, such as `~x` for a fresh variable, `$x` for a public variable, `#i` for a temporal variable and `m` for a message. Moreover, a string constant `'c'` denotes a public name in pub, which is a fixed, global constant.

NSPK's first message $A \rightarrow B : \{N_A, A\}_{pkB}$ is modelled in Tamarin as shown in Listing 2.1.

```
rule A_1:
  let m1 = aenc{'1', ~nA, $A}pkB in
    [ Fr(~nA) , !Pk($B, pkB) ]
  --[ OUT_A_1(m1)]->
    [ Out( m1 ) , St_A_1($A, $B, ~nA) ]
```

Listing 2.1 First message of the Needham-Schroeder Public Key protocol

The rule uses the fixed unary fact symbol `Fr` in the left-hand side to generate a fresh nonce (the symbol `~` denotes a fresh value). The rule uses the let-binding `let ... in` expression offered by the Tamarin syntax, which can be used to specify local term macros within the context of a rule. The let-binding, in this example, is used to define the message `m1`. The message `m1` is built with: *(i)* a constant `'1'`, *(ii)* the nonce `~nA`, *(iii)* the name of the agent `$A` who generated the message. The message is then encrypted using an asymmetric-key `pkB`, which is the public key of the agent `$B` . To send the message `m1` to the network, the fixed unary fact symbol `Out` is used in the right-hand side. To conclude, a state fact `St_A_1`, which represents the state of the agent `$A` (basically the knowledge of the agent at that time), is defined, containing the agent name `$A`, the other agent participating in the rule `$B` and the fresh nonce generated `~nA`.

The full NSPK protocol modelled in Tamarin is in Section A.1.

Restrictions. It is possible to restrict the set of traces to be considered in the protocol analysis taking advantage of restrictions. Since there are no restriction specified in the NSPK protocol, I can consider a simple authentication protocol, where an agent `A` sends a signed message to an agent `B` as shown, using the Agent `B` rule (and omitting the Agent `A` rule), in Listing 2.2:

```
rule B_1_receive:
  [ !Ltk(B, ltkB)
  , !Pk(A, pkA)
  , In(<m,sig>)
  ]
--[ Recv(B, m)
  , Eq(verify(sig,m,pkA),true)
```

```
, Authentic(A,m), Honest(B), Honest(A) ]->
[ St_B_1(B, ltkB, pkA, A, m)
]
```

Listing 2.2 Agent B receives first message

Here, agent `B` verifies the signature `sig` on the received message `m`, checking that the message `m` was really signed using the private key associate with the public key `pkA`. The result of the `verify` function is `true` or `false`. In this example, the function `Eq` is used to check that the result of the `verify` function to the received message is the constant `true`. The restriction `Equality` uses the function `Eq` to restricts the traces considered by the protocol only to those in which the equivalence is verified. This is achieved in the restriction imposing that the values `x` and `y` in the function `Eq` will be the same, as shown in Listing 2.3.

```
restriction Equality:
 "All x y #i. Eq(x,y) @i ==> x = y"
```

Listing 2.3 Equality restriction on two messages

Properties. Desired security properties of the protocol are defined over traces of the action facts of a protocol execution. Each label of action section must therefore contain enough information to state the properties. In Tamarin, properties to be evaluated are denoted by lemmas.

In Tamarin, a rule can be applied to a state if it can be instantiated such that its left-hand side is contained in the current state. In this case, the left-hand side facts are removed from the state, and replaced by the instantiated right-hand side. The

application of the rule is recorded in the trace by appending the instantiated action facts to the trace. All action fact symbols may be used in lemmas.

Tamarin provides the follow syntax for specifying security properties:

- `All` for universal quantification, temporal variables are prefixed with #;
- `Ex` for existential quantification, temporal variables are prefixed with #;
- `==>` for implication;
- `&` for conjunction;
- `|` for disjunction;
- `not` for negation;
- `f @ i` for action constraints, the sort prefix for the temporal variable 'i' is optional;
- `i < j` for temporal ordering, the sort prefix for the temporal variables 'i' and 'j' is optional;
- `#i = #j` for an equality between temporal variables 'i' and 'j';
- `x = y` for an equality between message variables 'x' and 'y'.

NSPK's security property is modelled in Tamarin using *injective agreement* as shown in Listing 2.1. A protocol guarantees to an agent `agent` injective agreement with an agent `peer` on a message `params` if, whenever `agent` completes a run of the protocol, apparently with `peer` in role `peer`, then `peer` has previously been running the protocol, apparently with `agent`, and `peer` was acting in role `peer` in his run, and the two agents agreed on the message `params`. Additionally, there is a unique matching partner instance for each completed run of an agent, i.e., for each Commit by an agent there is a unique Running by the supposed partner.

To analyse a protocol with respect to this property, an appropriate rule in role `A` with a `Commit(a,b,<'A','B',t>)` action and in role `B` with a `Running(b,a,<'A','B',t>)` action have been labelled. Here `a` and `b` are the agent names (public constants) of roles `A` and `B`, respectively and `t` is a term.

```
// Injective agreement from the perspective of
both the initiator and the responder.
lemma injective_agree:
  " /* Whenever somebody commits to running a session,
   then*/
    All agent peer params #i.
        Commit(agent, peer, params) @ i
      ==>
        /* there is somebody running a session
        with the same parameters */
          (Ex #j. Running(agent, peer, params) @ j & j < i
            /* and there is no other commit
            on the same parameters */
            & not(Ex agent2 peer2 #i2.
                    Commit(agent2, peer2, params) @ i2
                    & not(#i = #i2)
                  )
          )
        /* or the attacker perform a long-term key reveal
        on agent or peer */
        | (Ex #r. RevLtk(agent) @ r)
        | (Ex #r. RevLtk(peer)  @ r)
```

```
"
```

Listing 2.4 Example of property of the Needham-Schroeder Public Key protocol

In order to prove if the lemmas are correct, I need to execute the Tamarin prover from the command line. A browser-based window will be shown as in Figure 2.5.

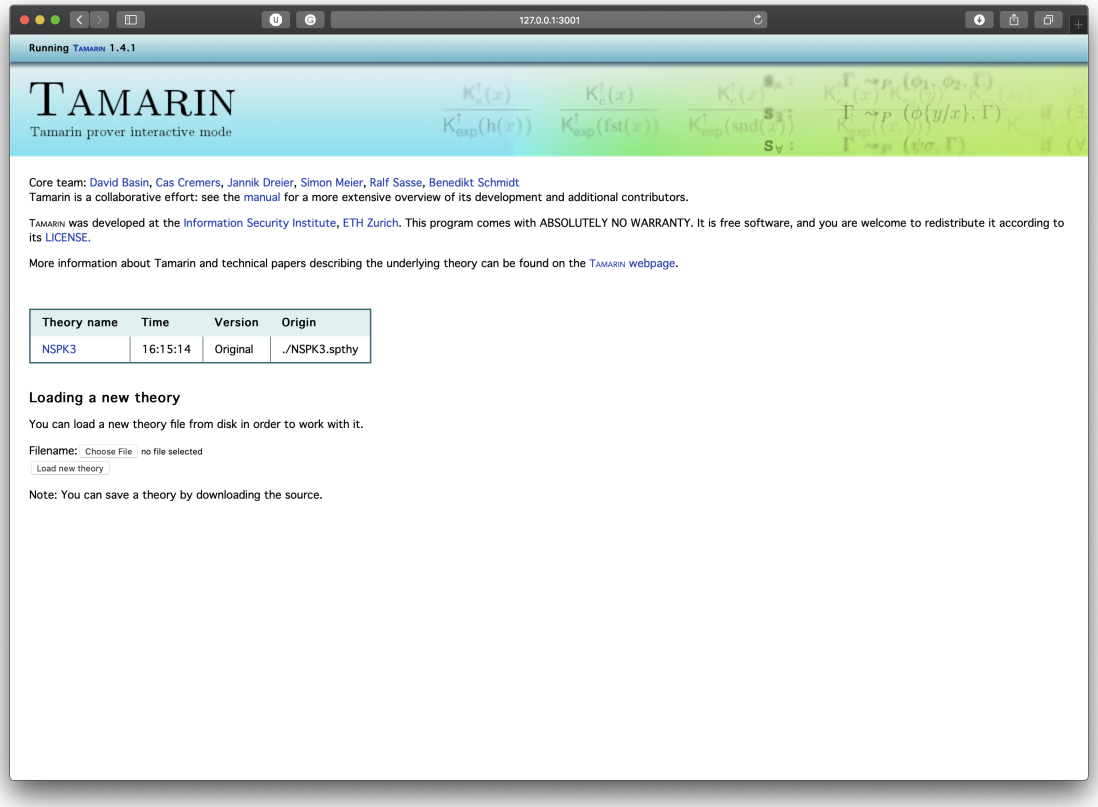


Figure 2.5 The GUI of Tamarin

The table in the middle shows all loaded theories. It is possible to either click on a theory to explore it and prove the security properties, or upload further theories using the upload form at the bottom.

By clicking on the “NSPK3” entry in the table of loaded theories, a new window appears, showing the theory on the left-hand side, with the links to the message theory describing the attacker, the multiset rewrite rules and restrictions describing

the protocol, and the raw and refined sources, followed by the lemmas to prove. This is shown in Figure 2.6.

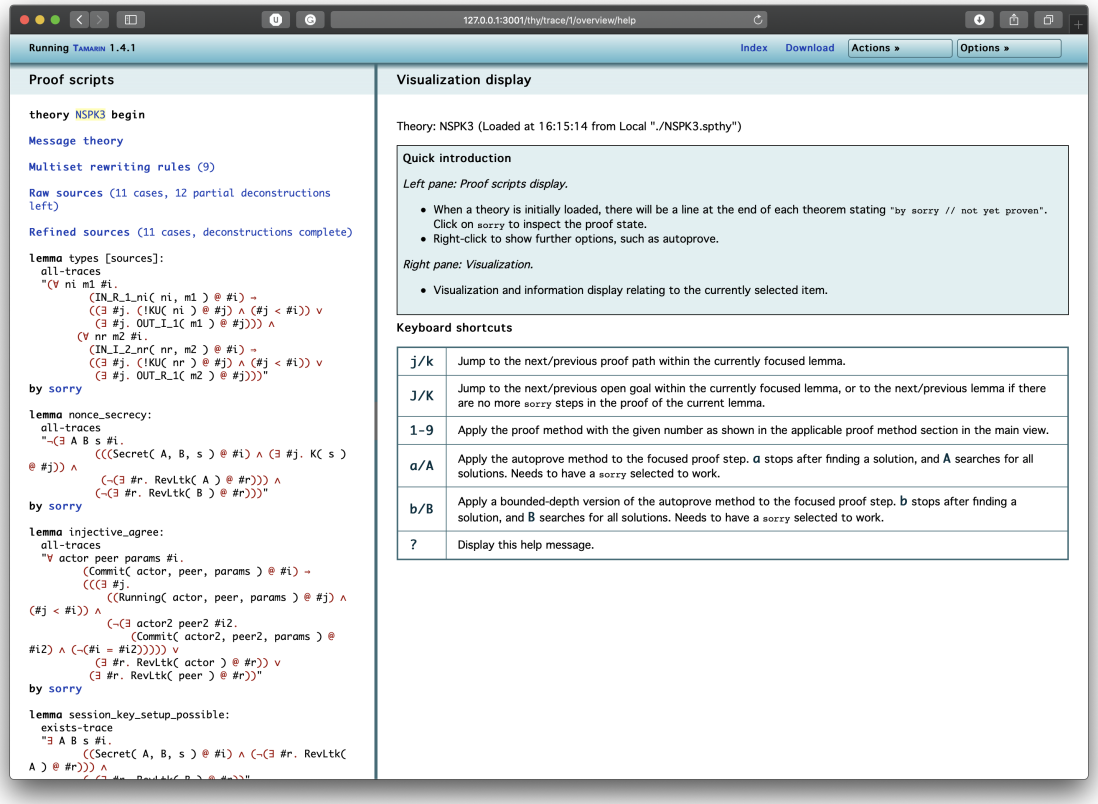


Figure 2.6 The NSPK3 model displayed in Tamarin

By clicking on the label “sorry” on a specific lemma, it is possible to see the applicable proof methods for that specific lemma, as shown in Figure 2.7. Tamarin proves lemmas using constraint solving. Namely, it refines the knowledge it has about the property and the protocol until it can either conclude that the property holds in all possible cases, or until it finds a counterexample to the lemma.

By clicking on the command ‘autoprove’, Tamarin selects the next steps for the proof, based on a heuristic. The lemma is coloured in green if it is successfully proven. If Tamarin finds a counterexample, it is coloured in red, as shown in Figure 2.8.

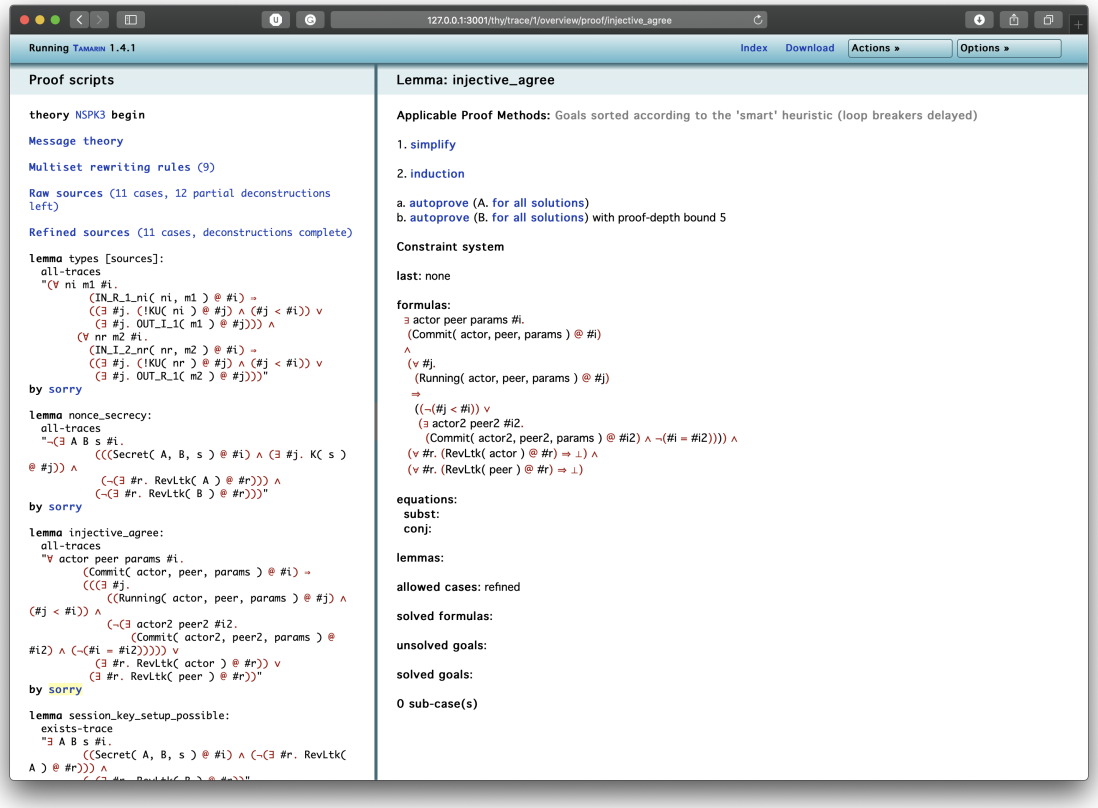


Figure 2.7 The proof methods for the NSPK3 lemma

Tamarin has been chosen for this work because it is one of most powerful tools for security protocol analysis and, in particular, provides support for cryptography, cryptographic channels and intruder capabilities. However, my approach is general and independent of Tamarin and could be applied similarly to other tools, even though once a tool has been chosen, it is typically necessary to adapt the approach to that specific tool. A short discussion about how to provide reusability of my approach with other tools is in Section 5.8.6.

Now we have all the necessary background to understand the following chapters. Wherever needed, I will provide additional information (e.g., about Tamarin, etc.).

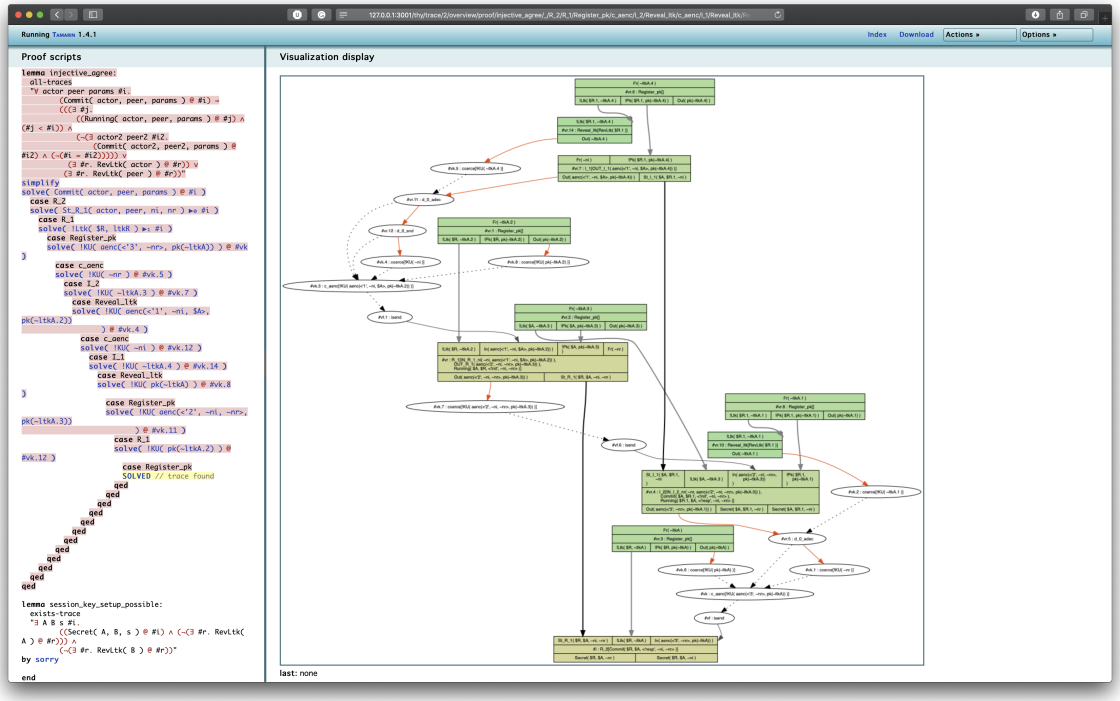


Figure 2.8 The counterexample found of NSPK3 highlighted in red

Chapter 3

An Investigation into the "Beautification" of Security Ceremonies

In this chapter, I introduce a study that I carried out to understand how to "beautify" security ceremonies. The work described here is based on the paper: Bella, G., Renaud, K., Sempreboni, D., Viganò, L. (2019). An Investigation into the "beautification" of Security Ceremonies. In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE - Volume 2: SECRYPT*, pages 125–136. Scitepress Digital Library. [25].

3.1 Motivation

Security measures can trigger unintended and unanticipated side effects if users consider them unattractive. By unattractive, I mean difficult, arduous, inconvenient and generally a nuisance. If people perceive their interactions with security systems negatively, instead of something that is there to protect them and their data, they

might try to bypass or game them (for instance, by using weak passwords). People will dread an encounter with a system they consider unattractive. Yet "beauty" is likely to be in the mind of the beholder, and it is not necessarily obvious how to beautify security systems.

Bella and Viganò introduced the *beautiful security approach* [26], postulating that security should:

1. become a primary, inherent feature of the system; *at the same time*,
2. not be disjoint from the system functionalities; *at the same time*,
3. contribute to the very positive experience that the user has of the system, ultimately making that experience beautiful.

While the first two are dealt with elsewhere, the third is the one that this chapter is concerned with: how can security ceremonies *contribute to* the positive experience that a user has of a secured system, ultimately triggering a perception of beauty?

Because users are an essential and integral part of the greater socio-technical system, they have to engage with *security ceremonies* [58], which are added to systems in order to secure them.

Users engage in a kind of 'ritual' with security ceremonies, with predetermined actions being actioned by the two parties in a prescribed order when they interact with the ceremony. The result may be that interaction with the security ceremony is far from straightforward. The need to ensure the security of systems may unintentionally make interaction with security ceremonies complex (and possibly unattractive).

As remarked above in Chapter 2, a number of approaches have been developed to inform the design and analysis of secure ceremonies, e.g., [20, 131, 88, 104], but these seem to neglect the notion of beauty. I coin the term "*beautification*" to refer to the process of making security ceremonies (more) beautiful. I know that beautifying

existing ceremonies by merely simplifying them has sometimes led to the deployment of insecure ceremonies. Such ceremonies contain vulnerabilities that could be exploited by an attacker. A prime example is the use of fallback questions to allow people to recover from forgotten passwords, but which often serve as a convenient back door for attackers [141]. In sum, it is known that security ceremonies have functional goals that guarantee requisite security properties. For example, a password recovery ceremony must ensure confidentiality of the replacement password.

This need motivates the need for a methodology that makes beautiful security practically applicable.

3.2 Contributions

The main contribution of this chapter is thus *the definition of a methodology that makes beautiful security practically applicable through new general guidelines.* More specifically, I provide four concrete contributions.

1. Preliminary investigation I leveraged crowdsourcing to help me to orient security ceremony design towards greater beauty. Answering the question "what constitutes a beautiful user experience of a security ceremony?" Bella and Viganò [26] asked is far from trivial both because of the vastness of the spectrum of possible answers and because, as researchers working in security, our own point of view is likely to be biased. For these reasons, I decided to source an answer to this question from a heterogeneous global crowd of people by means of a questionnaire.

2. Definition of guidelines The insights of the questionnaire have been generalised as a set of four guidelines. Each guideline reflects the result of a single question, which in turn reflects how people characterise the beauty of security ceremonies.

3. Operationalisation of the guidelines As proof of concept, I apply the guidelines to four real-world security ceremonies aimed, respectively, at voting, logging into a computer, setting up a password and entering physical premises. The result of this phase is the generation of new security ceremonies that respect the guidelines.

4. Final investigation Once again, I leveraged crowdsourcing to help me to determine whether applying my guidelines to ceremonies, or highlighting improvements based on my guidelines in existing ceremonies, had been effective. My findings are promising but require careful interpretation, as I shall explain. Furthermore, I address, through a semi-formal analysis, the security concerns regarding the beautification process.

3.3 Outline

A brief review of the relevant literature is in Section 3.4. My approach to consulting crowdsourcing platform participants using questions aimed at gathering their views on what makes cybersecurity beautiful is in Section 3.5. Taking the results in Section 3.5, beautifying guidelines that have been formulated to inform the design of more beautiful ceremonies are in Section 3.6. Application of the guidelines listed in Section 3.6 to existing ceremonies, attempting to make them more beautiful is in Section 3.7. Further consultation to crowdsourcing platform participants to determine whether the intended beautification of the ceremonies was successful is in Section 3.8. Security concerns are tackled in Section 3.9. Conclusions are discussed in Section 3.10.

3.4 Beauty in Security

Beauty is traditionally thought of as something visual, but it has also been applied to music [128], mathematics [60, 135], truth [116] and design [67].

The experiments carried out in [168] indicated that a strong correlation between a system's perceived aesthetics and its perceived usability. The study highlights that users are capable of distinguishing between various properties of a system that can potentially contribute to make the perception of that system "beautiful", leaving room for me to consider security as one of those properties. Findings of this study seem to suggest that the use of a system does not affect the perception of it, however further research [169] suggests that this initial perception can be changed through use of the actual system. This study seems to corroborate two aspects: *(i)* there is space to consider the correlation between usability and beauty, *(ii)* beauty is interwoven with a person's experiences. Hassenzahl et al. [75], however, have criticised *(i)*, showing that the relationship between beauty and usability is mediated by goodness.

Carritt [38] supports *(ii)*, claiming that beauty is not simply related to the agreeableness or usefulness of an item or experience. He says that beauty has more to do with a contemplation of a feeling experienced during, or remembered after, an encounter with a particular artefact.

What characteristics of this experience might contribute to beauty? The literature suggests the following: ease of use (fluency) [132], a sense of pleasure [162], simplicity [41, 89, 69], aptness [67], elegance [67], the value of the system in a given context (goodness) [75] and responsiveness [121]. This review suggests that beauty will be attributed to an artefact if it elicits positive feelings, based on prior user experiences.

Let me now consider the idea of *beautiful* security software. This is especially important because its non-use or, worse, a negative experience of an unattractive ceremony, will compromise security and leave holes open for hackers to exploit. Current experiences

of common security software appear to confirm their general unattractiveness [47, 43]. The consequence is that users might act to circumvent the ritual [31]. Awareness and training programmes are the standard organisational response to this [179], but the effectiveness of such drives is patchy [11, 91]. Training does not work because it can not overcome a reluctance that stems from previous negative experiences with unattractive software.

What I am proposing is to *beautify* security ceremonies so that people will *want* to use them because the beauty thereof results in positive feelings. What I seek to discover is exactly how to achieve this. In summary, I want to build security ceremonies that users will want to engage with because doing so has been a positive experience. This might mean that the software displays one or more of the general characteristics listed above. It could also be that beautiful security ceremonies have their own set of characteristics: identifying these is the topic of this work.

3.5 Crowdsourcing for the Concept of Beautiful Security

Bella and Viganò [26] asked a fundamental question: what constitutes a beautiful user experience of a security ceremony? Answering this question is far from trivial both because of the vastness of the spectrum of possible answers and because, as researchers working in security, our own point of view is likely to be biased. For these reasons, I decided to source an answer to this question from a heterogeneous global crowd of people by means of a questionnaire.

My questionnaire aimed to gain an understanding of how the users of security ceremonies perceive and describe them by means of a scale of "*emotional*" values, leaving them free to express as many values as they desired. In other words, I designed

the questionnaire to explore a correlation between security ceremonies and how their beauty is sensed. The questionnaire consisted of the following four questions:

Q1 Think about when you log into your bank account, and you use one of those little devices the bank sends you that displays a number for you to enter as your password. Do you consider this process to be: *Fun, Beautiful, Excessive, Annoying, Engaging, Essential, Reassuring, Appealing*?

Q2 When you were a child, did you ever read a book about a group of children having a secret club, with meetings where people were allowed in when they knew the secret password? In this case, how would you rate the security of this process in terms of: *Fun, Beautiful, Excessive, Annoying, Engaging, Essential, Reassuring, Appealing*?

Q3 Think about a web-based system that requires you to provide a password that meets certain rules (e.g., upper case, lower case, digit, special character, minimum length). In this case, how would you rate the security in terms of: *Fun, Beautiful, Excessive, Annoying, Engaging, Essential, Reassuring, Appealing*?

Q4 Think about old-fashioned burglar alarms that used to go off every time a bird landed on the roof, or lightning struck nearby. The only way to correct a false alarm was to get home as quickly as possible to enter the PIN to shut it up. Modern systems are different. They are often controlled from your smartphone. Now you can keep an eye on them from wherever you are in the world, and even see if everything at home is in order, by linking to cameras in your home. False alarms can quickly and easily be dealt with, without annoying your neighbours for hours on end. This is an example of how physical security has become more engaging. *Can you think of a way that cybersecurity could improve in the same way*?

The first three questions were closed-ended, allowing people to respond with multiple preferences (and thus allowing me to gain insights into their feelings, where these could not be represented by only one choice). The final question was open-ended. The rationale behind the questions is as follows.

3.5.1 Q1 — Token Device

The first question concerns the technology that people use to confirm their identity upon login to their own online bank accounts. Originally, when a customer activated the e-banking functionalities on their bank account, the bank would supply them with a plastic card or a sheet of paper with several *one-time passwords (OTPs)* to enable them to carry out two-factor authentication[1], in this case using their username-password pair as well as an OTP. More recently, banks started handing out small electronic devices to their customers that are capable of generating an OTP on the fly, instead of their having to consult a pre-printed list of fixed OTPs. Nowadays, these devices are very common[2].

I devised the first question to explore reactions to OTPs as a mechanism that is widely used to prevent bank-account hacking but that is sometimes not as usable as it should be [159]. OTP devices are intended to help users to handle a situation in which security ought to be the first goal, even though the way it achieves this might not be ideal. I wanted to find out what respondents thought about needing to possess such a physical device to generate OTPs. This question allowed me to gather evidence to

[1]Two-factor authentication asks the user who wishes to be authenticated to provide two different secrets: something the user knows (e.g., a password), something the user has (e.g., a device), something the user is (e.g., unique biometric features such as fingerprints or the patterns on a person's retina blood vessels). This can be straightforwardly extended to multi-factor authentication, where the user has to provide multiple different authentication factors.

[2]In fact, mobile applications have been developed to enable users to generate OTPs on their smartphones, thus replacing also these small electronic devices. However, given that these mobile applications are not yet widespread (only some countries and some banks have adopted them so far), I decided not to include this option in my questionnaire.

understand whether current two-factor authentication ceremonies are in line with my ideas or whether I ought to start thinking about alternative approaches.

3.5.2 Q2 — Secret Club

The second question is about using an episode of the British pre-school animated television series "Peppa Pig" [26]. In this episode, titled "The Secret Club", Peppa's friend Suzy Sheep is building a secret club whose membership is identified by wearing a mask. Joining the club demands uttering a one-time password, as shown by the following excerpt:

Peppa: *Hello, Suzy.*

Suzy: *Hello, Peppa.*

Peppa: *Why have you got that mask on your face?*

Suzy: *So people don't know it's me. I'm in a secret club.*

Peppa: *Wow! Can I be in your secret club?*

Suzy: *Shh! It's not easy to get into. You have to say the secret word.*

Peppa: *What word?*

Suzy: *Blaba double!*

Peppa: *Blaba double!*

Suzy: *Right, you're in.*

Later on Suzy will comment that the password *changes all the time to keep it secret!* It would appear that anything related to secrecy is perceived to be an enjoyable game, even before the rules are set, and Bella and Viganò use this to propose that a user's experience of a security ceremony could be appealing and beautiful as is demonstrated by the childish enthusiasm with which Peppa reacts in this episode.

What would my respondents think of a similar situation? Even though it could seem a fairly childish example, I devised the second question to help the respondents connect with similar experiences from their own childhoods, thereby allowing me to determine whether the respondents agreed with Bella and Viganò's assessment of the attractiveness of this example. Thus, this second question aimed to give me a reference point for the kind of user experience that I would like to achieve when security is seen in terms of beauty and attractiveness.

3.5.3 Q3 — Password Creation

Given that the second question considered passwords in a childish context, I devised the subsequent question to ask about passwords in a more adult context, thus also allowing me to check whether the answers obtained would be in line with those given in response to Q2. More specifically, Q3 considers a thorny situation that users have to face at least once a day. A cloud service, an email account, a favourite e-commerce website, or maybe a laptop account, all require the user to enter a secret password correctly to gain access. Nowadays, this usually requires a strong password that respects certain criteria (such as a minimum length, different characters, special characters) and is also semantically different from the user's sensitive information (such as name or birth date). I asked respondents to express how they felt about engaging with this particular security ceremony. I expected to obtain results that would reflect the users' frustration with current password management systems.

3.5.4 Q4 — Cybersecurity Improvement

I chose to conclude with an open question to capture the respondents' different points of view. Cybersecurity will continue to evolve and I believe that this evolution could take many different directions. I devised the final question to collect a wide variety of

suggestions about the different ways of making security ceremonies more attractive, and thus of pursuing beauty. By using an analogy with an old-fashioned burglar alarm, I asked respondents to assess the attractiveness of modern solutions and to propose improvements related to cybersecurity.

3.5.5 Findings

I administered the questionnaire to one hundred respondents. I used the CrowdFlower platform and did not constrain respondents in terms of language or geography. CrowdFlower workers span the globe and heterogeneity of the sample is assumed. The results for the first three questions are summarised in Figure 3.1, whereas Table 3.1 shows the results of the answers to Q4.

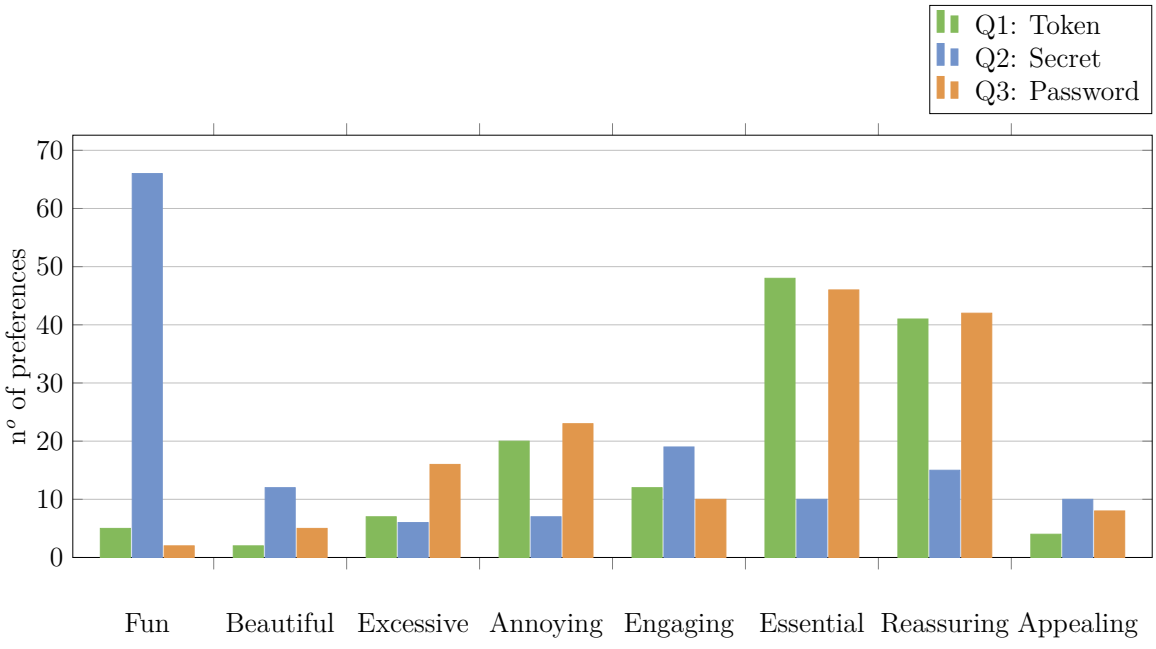


Figure 3.1 The answers to questions Q1, Q2 and Q3

Findings from Q1, Q2 and Q3

The answers to Q1 show that users most often perceive these kinds of ceremonies to be *Essential* (48) and *Reassuring* (41). The third most popular reaction is that engaging with the ceremony is *Annoying* (20), confirming that users find it frustrating. The least chosen options are: *Fun* (5), *Appealing* (4) and *Beautiful* (2). This reinforces my initial assertions about the unattractiveness of current security ceremonies.

In Q2, users largely chose *Fun* (66), followed by *Engaging* (19) and *Beautiful* (12), and finally *Annoying* (7) and *Excessive* (6). This could mean that the users see that the situation is less serious but also that they perceive this to be less burdensome than rigorous security ceremonies. This is so even though the secret mentioned in the cartoon is very similar to an actual password.

In Q3, we can observe similar responses to Q1. I obtained a high number of preferences for *Essential* (46) and *Reassuring*(42), reflecting the fact that users sense the importance of a tangible feeling of reassurance. Similar to Q1, the third most voted response is *Annoying* (23) followed by *Excessive* (16). Very few respondents considered the password management to be *Fun* (2) or *Beautiful* (5).

What we can understand from the responses is that the respondents seem to grasp the *Essential* nature of security ceremonies, and they also feel *Reassured* by security measures such as passwords and mobile tokens. However, they undoubtedly feel annoyed and consider the security measures *Excessive*, especially for password-like ceremonies. The burden of having external devices such as token generators or complicated passwords could be responsible for the number of *Annoying* responses. In designing more beautiful security ceremonies I have to take this into account.

If we focus on the *Fun* and *Beautiful* aspects, I can see that a majority of the users believe that having a secret code in a childish context is funny but also beautiful and *Engaging*, as compared to the other two contexts. In Q2, I did not mention details like

the complexity of the secret but, even so, the perceived positivity of this Peppa Pig experience reveals the negativity of existing real-life security measures.

If we return to what Carritt [38] says, these negative experiences and perceptions will lead to users not considering the engagement with security ceremonies to be "beautiful".

Findings from Q4

In Q4, I analysed the 100 open-ended question responses. From an open-ended question I expected several possible kinds of answer, which I had not only to check but also to classify. In questions like this, some respondents just provide a random answer as they are not interested, or are too lazy, to think and provide a meaningful answer. My analysis approach was the following. First of all, I skimmed the list for yes/no/I-don't-know answers, removing 27 answers. After that, I removed 32 answers that I deemed to be out of context (e.g., due to apparent misunderstandings). I classified and grouped the 41 remaining answers based on the general idea or suggestions that they communicated.

The results are shown in Table 3.1. The general categories that emerged from the analysis are: *SMS*, *Geolocation*, *Biometric*, *Alert* and *Password*. *SMS* suggests to use text messages to warn the users that something is happening. *Geolocation* suggests to use the GPS position or Wi-Fi location to prove that some operations are allowed from the place where they are executed. *Biometric* suggests the use of fingerprint or face recognition, or other biometric-recognition technology to authenticate the users. *Alert* suggests the use of some kind of notification (e.g., push notifications) to alert the users. *Password* suggests that passwords be used, but that they be improved in different ways (e.g., reducing the number of characters, relaxing some of the constraints on character type, or the "distance" from the previous password).

SMS	**Location**	**Biometric**	**Alert**	**Password**
7	2	17	8	7

Table 3.1 Categories for the answers to question Q4

Biometric technology was suggested by 17 respondents as a viable way to beautify the security experience, thanks in particular to the recent spread of fingerprint and face recognition facilitated by smartphone manufacturers. 8 respondents expressed the will to improve cybersecurity by using some kind of alerting system through notifications, as long as this is integrated into a two-factor or multi-factor authentication solution with multiple devices enabled to receive warnings.

New methods based on passwords and SMS-enabled technology were indicated by seven respondents. Finally, two respondents suggested using geolocation as a method that checks the position of devices to authorise certain operations.

3.6 Operationalising the Findings

The idea behind the questionnaire was to understand how people characterise the beauty of security ceremonies in such a way that I could formulate general guidelines to inform the design of beautiful security ceremonies, or of beautified versions of existing security ceremonies. The previous section discussed the answers to my questionnaire and reported the insights I gained. It is possible to extract a series of guidelines from these insights. These guidelines would represent a concrete link between the users *"emotional"* values as responses and the security aspects within the ceremonies considered in each of the questions. So, it is possible, for the four questions, to derive four guidelines.

G1. Feel unencumbered: you do not need to carry anything along.

G2. Leverage the sense of the group.

G3. Simplify the rules, even though they seem essential.

G4. Use biometrics techniques.

G1 enhances the sense of freedom and it comes from the insights originated from Q1. At some point, security analysts ruled that "possessing" something enables us to pursue security through what we hold. This changed the pre-existing password approach due to the perception that holding something is easier than remembering complicated password. Unfortunately, this is not always true. Human nature makes us prone to forgetting seldom-used objects and we also easily lose small objects. Cards and one-time token generators are good examples of objects that are easily lost or mislaid.

However, a ceremony that asks a user to prove that they hold an object might not be the best solution to the perceived unattractiveness of existing security ceremonies. The need to possess an essential security ceremony object could easily be considered burdensome. It can lead to frustration each time a user cannot engage with it. This is why I advance guideline G1 as a solution to this burden every time a person needs to demonstrate ownership of a physical object.

G2, which derives from the insights for Q2, is oriented at a ceremony design that makes the user feel part of a group. This situation is well represented in the Peppa Pig episode [126] where having a secret word implies being part of a secret club. This view was supported by the responses that I obtained in the questionnaire because the situation I described was perceived as being *Fun* by most respondents. Although it would seem unusual to achieve this level of agreement in a such diverse context, I want to strive towards exciting that same feeling of enthusiasm when people interact with systems securely.

The simple matter of having a password could be changed thanks to a change of context such as the episode but also in terms of the way you prove that you have knowledge of it. The idea of being enthused in some way could make the experience more positive. This is why we argue that guideline G2 could be a way to imbue a sense "charm" into security ceremonies, from the users' perspective.

G3, stemming from Q3, is a reasonable assumption based on the fact that having simpler ceremony rules improves the user's experience while preserving security. What G3 advances is an approach that urges a return to simpler rules, thereby freeing the users from any extra burdens in the execution of essential security ceremonies. Einstein famously said that things should be "*made as simple as possible, but no simpler*". This applies admirably to my security context.

G4 suggests using "something you are" in security ceremonies. The answers to Q1 and Q3 revealed that users considered "something you know", such as a password, as well as "something you have" burdensome. An alternative approach that might be more acceptable could be the use of biometrics, as highlighted in the responses to Q4. Although the use of biometrics is not exempt from critique as many users (and some developers of security solutions) are concerned about their privacy ramifications, a variety of different biometrics are available and they do appear to represent a reasonable compromise, also conveying a sense of play and of being "cool and modern" in achieving security.

If I am able to introduce G4 in ceremonies, I will likely obtain a positive response from the users.

3.7 Applying Beautifying Guidelines

Bella and Viganò [26] highlighted the crucial aspects of security ceremonies to acknowledge the value of beautifying ceremonies. Developers can avoid their ceremony

becoming ineffective when people bypass it due to its unattractiveness. However, achieving cybersecurity requires systems to interact with many other pre-existing systems' ceremonies. Some of these are so part-and-parcel of others' systems that it might be difficult to replace them with more beautiful ceremonies built following my guidelines. I am concerned with the possibility of improving an existing security system with as little effort as possible. My intention is *(i)* to take a system that does not reflect the beautiful security approach [26] yet it is secure in the (often) optimistic assumption that the users will use it precisely as intended by the system designers, and *(ii)* to beautify the system following my guidelines so that this assumption becomes less optimistic and more realistic.

I consider four ceremonies that are, I believe, in dire need of beautification:

1. the Italian voting ceremony,
2. the Laptop login ceremony,
3. the Password setup ceremony, and
4. the European (EU) premises access ceremony.

I discuss ways to make such ceremonies more beautiful and for each of them I present a specific beautified version that I have devised, with the exception of the improved Laptop login ceremony that has already been developed by Apple.

The beautification changes that I propose reflect the guidelines I identified in the previous section (and also reflect the authors' combined experience of designing *Socio-Technical Systems*). In the following section, I will then report on the second questionnaire that I issued to assess whether the new versions of the security ceremonies that I present here are indeed perceived to have been beautified.

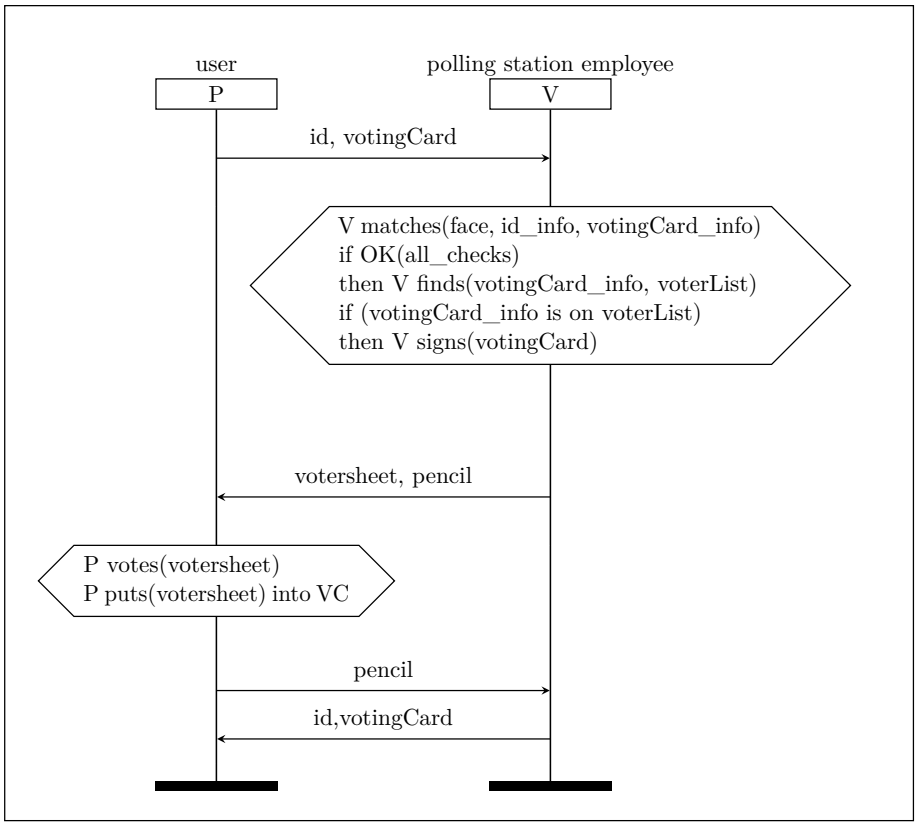


Figure 3.2 The current Italian Voting Ceremony

3.7.1 Italian Voting Ceremony

Figure 3.2 shows the voting ceremony currently used in elections in Italy. Here, and in the following, I represent security ceremonies by means of *Message Sequence Charts* (*MSCs*), which show the messages exchanged by the principals (agents) participating in the ceremony, as well as the internal computation that is performed by the principals. The MSCs that I consider in this work are, hopefully, quite self-explanatory.

In the Italian voting ceremony in Figure 3.2, there are only two principals: a user (the voter) and an employee of the polling station (the voting officer). This ceremony could be considered outdated in comparison with the voting ceremonies adopted in other countries (e.g., UK, Germany). Even if the UK voting ceremony seems to be more "hussle-free" since it does not require to carry anything along compared to the

Germany voting ceremony, which requires to take an ID card to go to vote, proceeding to make these ceremonies more "beautiful" is something that has to be done trying to minimise the changes in order to potentially reduce the introduction of security issues. The outdated part of the ceremony would be represented by the need of the voter to have two different documents:

1. a national ID card (or any other identification document such as a passport or a driving licence) and

2. a *votingCard*, issued by the city hall to record that a voter has cast a vote in this election.

This *votingCard*, which is shown in Figure 3.3, represents the ceremony's main sticking point for two primary reasons. Firstly, *votingCard*s are stamped at each election, and they fill up. If this happens, there are no more spaces for the voting officer to stamp the *votingCard* to register the voter's presence. A second problem occurs when a voter loses their *votingCard*. In both cases, the voter will first need to obtain a new *votingCard* from official sources at the city hall in order to be able to vote in this election (and some of the following ones, until the new *votingCard* is full).

During electoral rounds in the last years, these two issues have created particularly obvious difficulties, with long queues snaking out of city hall offices across Italy, populated by voters needing to obtain new copies of their *votingCard*s.

The complication comes from the use of a single document (the *votingCard*) for two different purposes: (1) a voting register and (2) an eligibility proof. Voters experience frustration, especially if the *votingCard*'s spaces have been filled up, in which case the *votingCard* only satisfies one of its two purposes: eligibility. Recrafting this ceremony offers me a clear-cut opportunity for improvement.

The new ceremony in Figure 3.4 applies what I have suggested with the beautification guidelines G1 "feel unencumbered: you do not need to carry anything along" and

Figure 3.3 *votingCard* currently used in Italian elections

G3 "simplify the rules, even though they seem essential". Italy, indeed, is among the stragglers with regard to digitalisation in Europe. Few improvements have been implemented recently on the e-health system, however, Italy is still far behind on the integration of digital technologies in the economy and Internet usage and this lack is pushed to the end user, making them carry several documents, some of them potentially redundant, in the case of the voting ceremony.

I suggest, in fact, to remove part of the burden voters see in having to use a *votingCard*, reshaping the ceremony and demanding one part of the security components of the voting ceremony that comes from this change to the polling station which is represented by its employees. Removing the "what you hold" object (the *votingCard* in this case) reflects the guideline G1 and, considering the voter's point of view, the ceremony should benefit of the changes reflecting also the guideline G3.

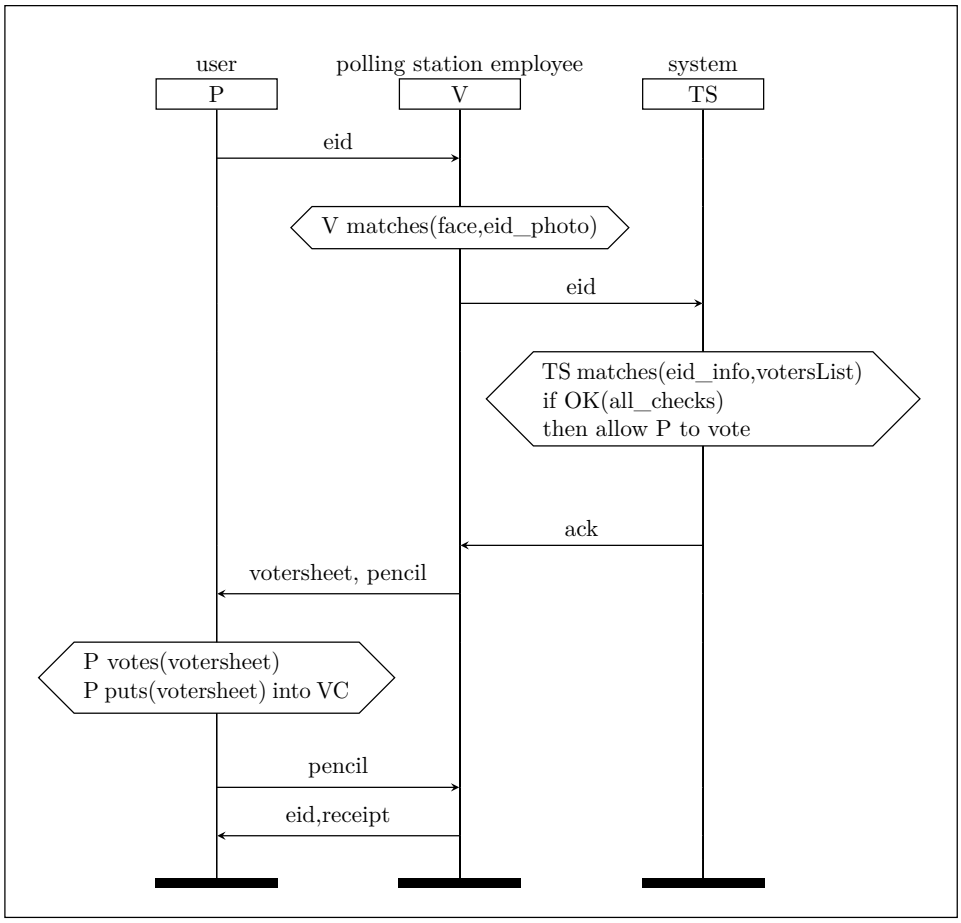


Figure 3.4 Potentially beautified Italian voting ceremony

The simplest way to replace the *votingCard* would be to use a system that needs the voter only to present their national ID card (or identification document), in its electronic form but also with the possibility to insert the data of the oldest ID card. The polling station employee would check the voter's eligibility and record their presence in a centralised database, by interacting online with a third principal in the ceremony (the "system"). The voter would no longer need to "hold" the *votingCard* that causes all the problems stated earlier. Overall, the beautification process here aims to make the user more willing to go to vote, trying to unburden the user and removing also some of the pitfalls that can happen due to the need for more voting documents.

Further improvements on the beautified version of the Italian voting ceremony as well as other voting ceremonies can be discussed, especially considering improvements that brings the voter to not carry anything along in addition to e-voting ceremonies. I leave this study as future work.

3.7.2 Laptop Login Ceremony

Every day, users have to enter multiple passwords into their laptops (and computers). These passwords are becoming longer and more complex over time due to the security requirements enforced by operating systems. Figure 3.5 shows "password entry", the login ceremony currently enforced by most laptop models.

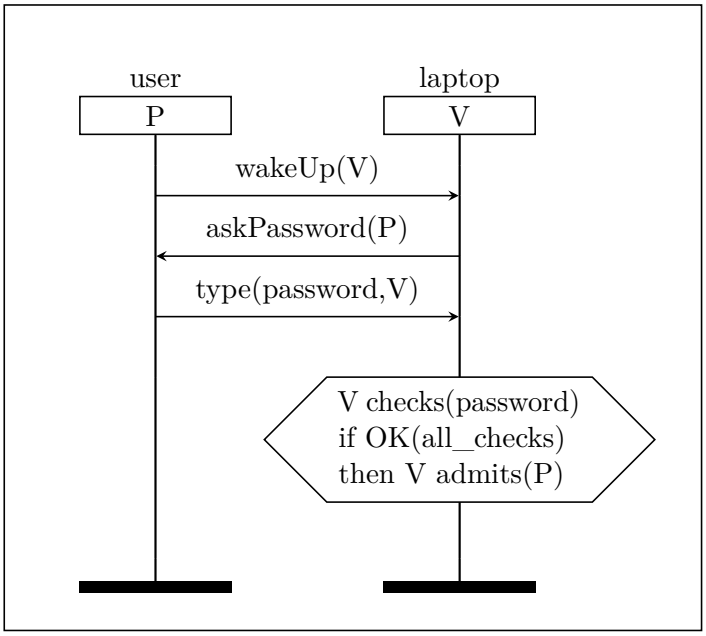


Figure 3.5 The widespread Laptop login ceremony

We can of course all agree that passwords are essential, but my questionnaire respondents confirmed my personal intuition (and my personal experience) that users increasingly dread this ceremony. To simplify the login, Apple has developed a new way

to login on its latest laptops by means of its Apple Watch product.[3] It is coherent with my guideline G2 if we see the Apple Watch as a distinctive feature of a group. This authentication method is incorporated in the beautified ceremony shown in Figure 3.7. Every time the user needs to wake up her MacBook, she just needs to press a key on the keyboard or touch the trackpad; after that, the system looks for a paired Apple Watch and if it finds one, the MacBook unlocks itself without requesting any other actions. A successful outcome is demonstrated in Figure 3.6. Similar ceremonies using one or more small portable/wearable devices have also been proposed [157].

Using the "something that I possess" paradigm here to replace the need of inserting the password, which reflects the "something that I know" paradigm, into the login ceremony, successfully validates the beautification guideline G2. Similarly to this case, the "something that I possess" is used in situations that require two-factor authentication (2fa) with hardware OTP tokens replaced by software OTP tokens generated by smartphone apps, taking also advantage of the pervasiveness of smartphones. In fact, people would feel as part of a group that uses a smartphone app, which seems more cool, instead of the old fashion OTP token device, to do banking operations.

3.7.3 Password Setup Ceremony

In 2003, the National Institute of Standards and Technology, published the NIST Special Publication 800-63 [70], where they suggested that users protect their accounts by inventing "awkward", new passwords that include upper case and lower case letters, numbers and special characters, and by changing their password regularly. Such passwords aim to be less guessable (e.g., more resistant to dictionary attacks). Nowadays, this is common practice, with websites and services forcing users to craft

[3]Google and Microsoft are similarly planning to remove passwords using the protocol Fido in conjunction with Android phones or physical tokens, respectively.

Figure 3.6 A successful MacBook login by Apple Watch

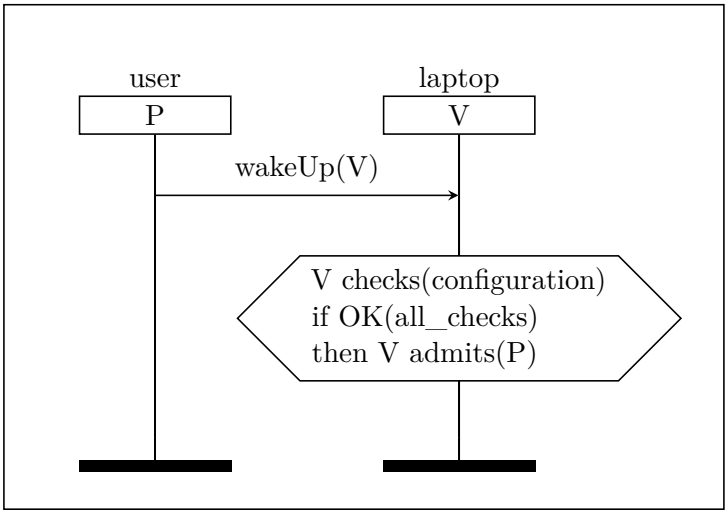


Figure 3.7 A potentially beautified Laptop login ceremony (by Apple Watch)

passwords respecting these constraints. Figure 3.8 shows a ceremony that reflects this practice.

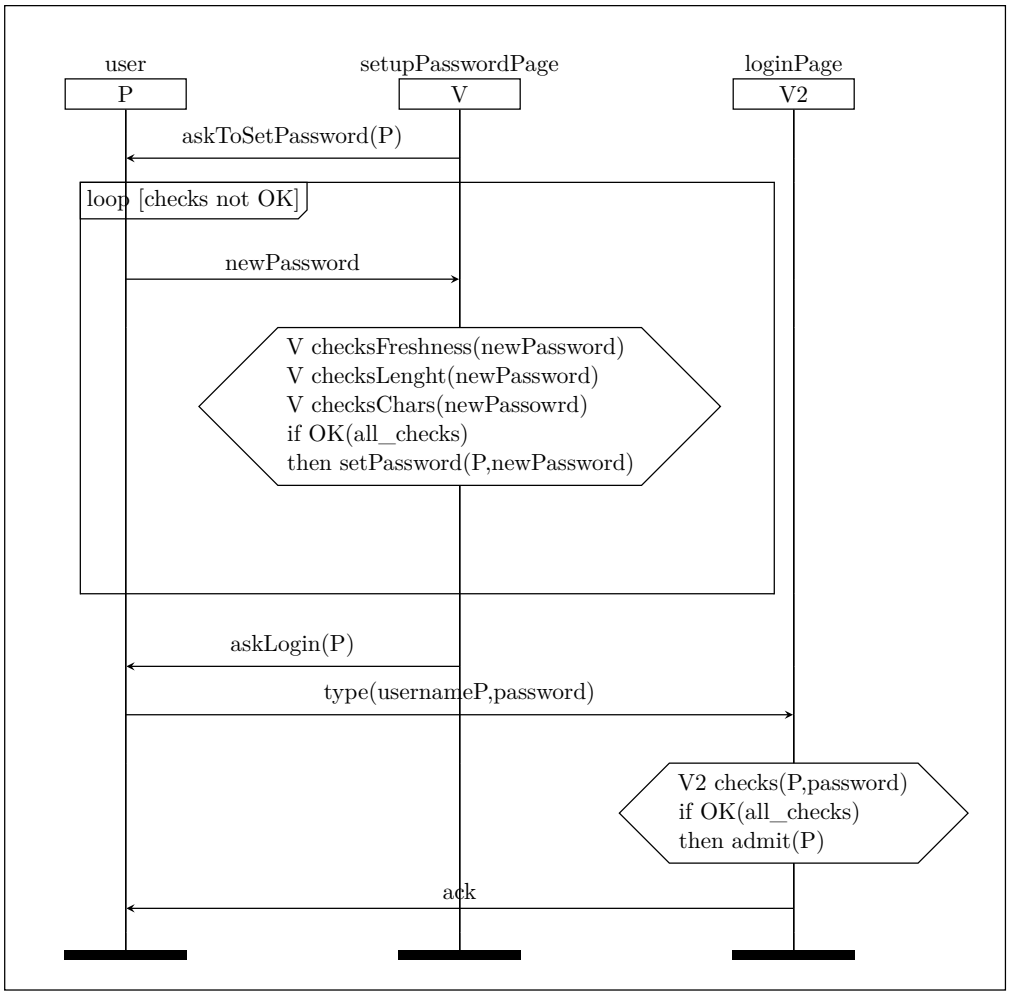


Figure 3.8 The Current Password Setup Ceremony (using NIST 2003 rules)

In 2017, Bill Burr, the person who originally proposed the rules, declared that he regretted publishing those strict rules [107]. In fact, the new NIST document published in June 2017 [70, Section 5.1.1.1] contains the following novel guidelines:

> *"...memorised secrets shall be at least 8 characters in length if chosen by the subscriber. Memorised secrets chosen randomly by the Credential Service Provider (CSP) or verifier shall be at least 6 characters in length and may be entirely numeric. If the CSP or verifier disallows a chosen memorised secret based on its appearance on a blacklist of compromised values, the*

subscriber shall be required to choose a different memorised secret. No other complexity requirements for memorised secrets should be imposed."

Applying these guidelines, it is possible to propose a new ceremony that allows users to choose whatever password they want, morphing the security component into a check on the number of times a password is typed incorrectly. By doing so, I simplify the rules imposed by the ceremony as suggested by G3. Figure 3.9 shows the new "Password setup" ceremony that I propose.

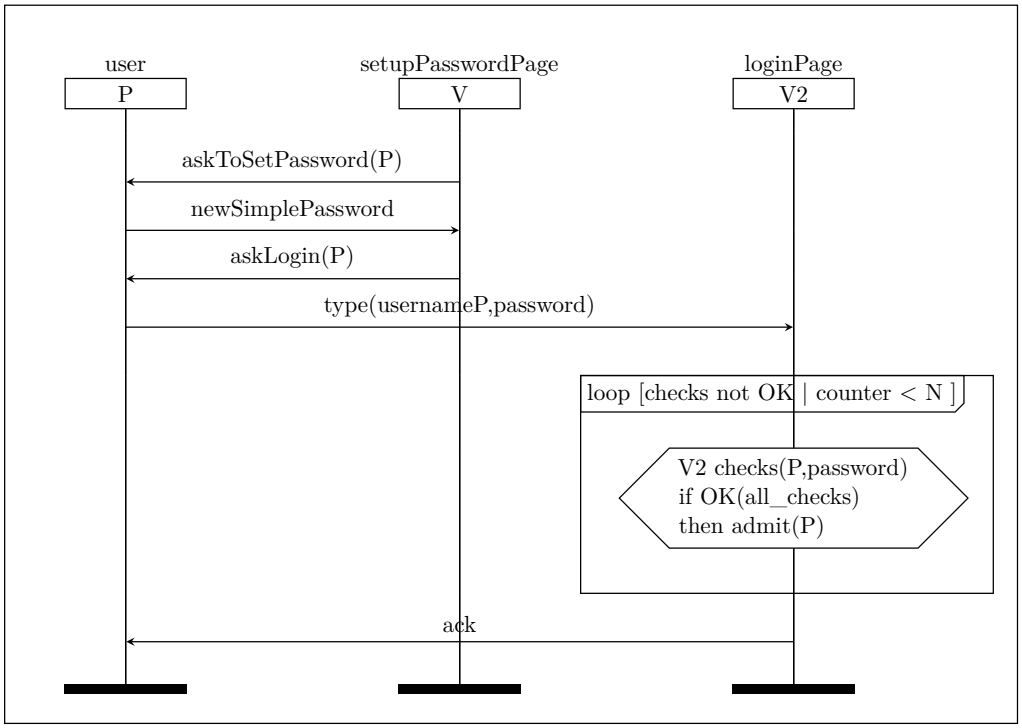


Figure 3.9 A potentially beautified Password setup ceremony (using the new NIST rules)

3.7.4 European (EU) Premises Access Ceremony

Those wishing to gain access to highly-secure buildings, such as some EU premises, have to engage with an authentication ceremony like the one shown in Figure 3.10, in

which the security staff check both a visitor's photo-ID (e.g., a national ID card or a passport) and their fingerprint, which is stored in a database.

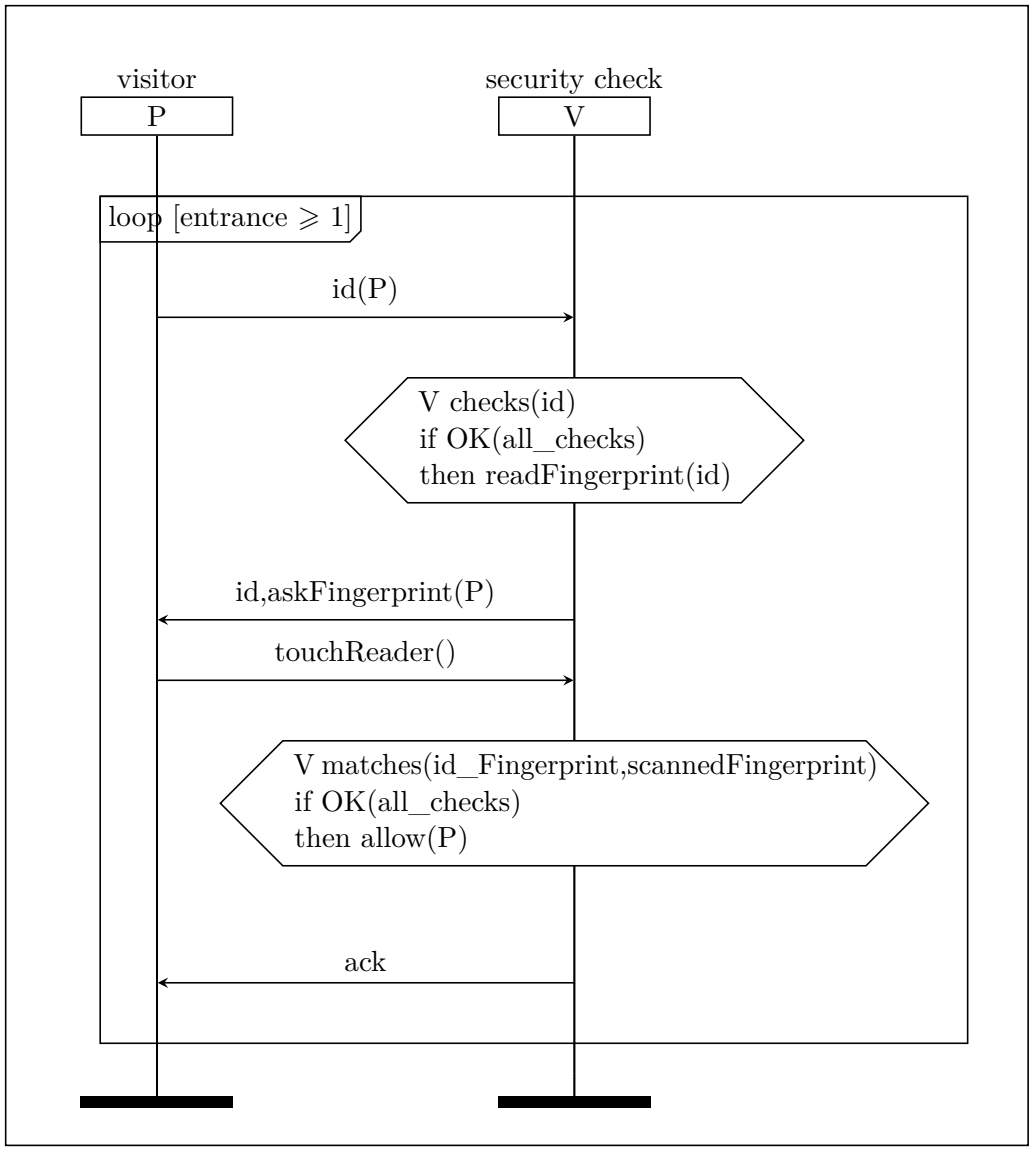


Figure 3.10 The current EU premises access ceremony

This ceremony thus requires a first-time visitor to engage in a time-consuming fingerprint-registration process. This ceremony is repetitive in the case of returning visitors, hence especially frustrating if a person needs to access the premises, for example, multiple times a day.

I can apply G4 to make the ceremony (more) beautiful. Still using biometric technology, it is possible to devise a leaner ceremony that requires the user to engage with fewer interactions. In this beautified ceremony, shown in Figure 3.11, I have removed the document check for the entrances after the first one, so that the main security checks are only addressed by means of fingerprint control.

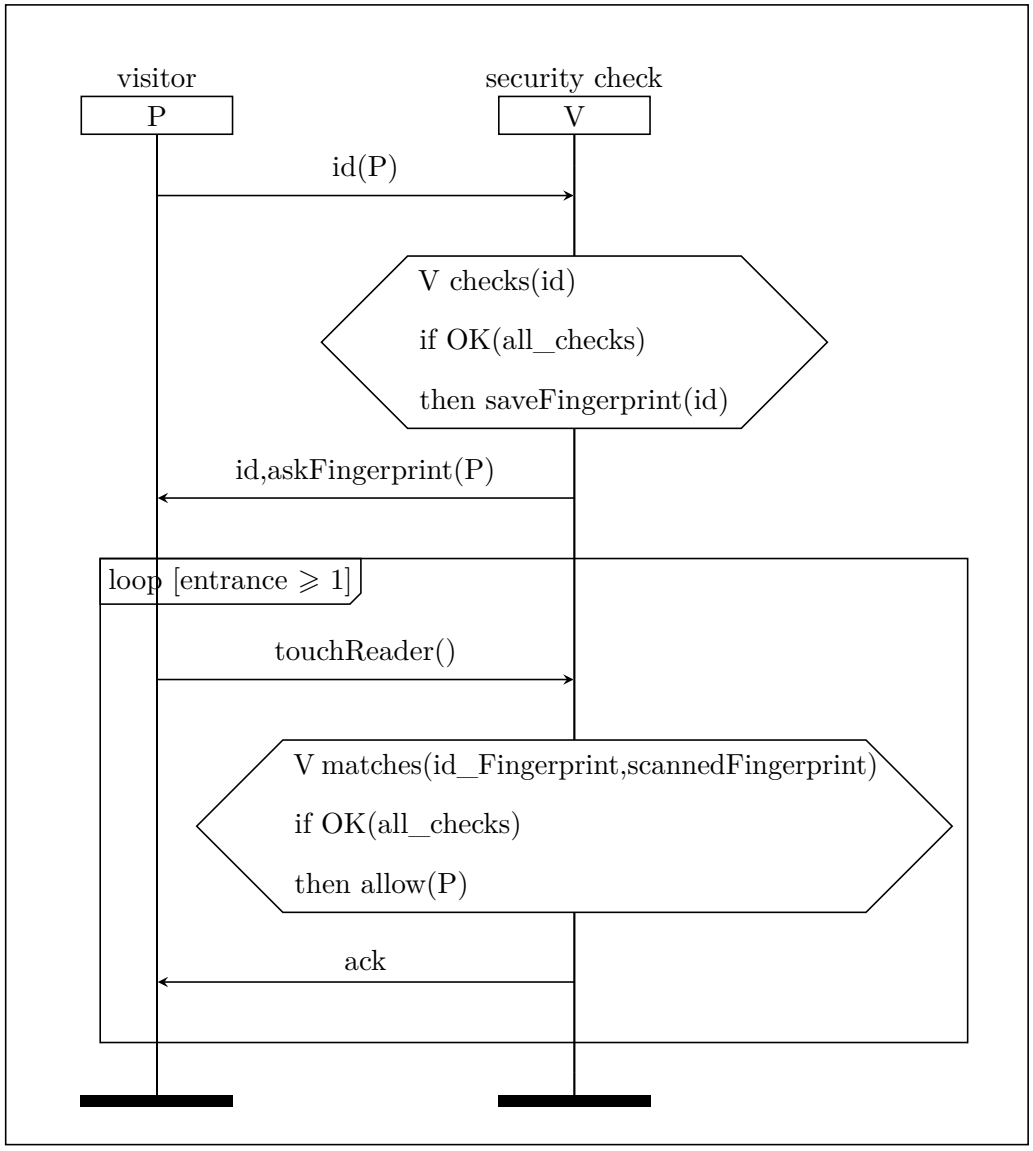


Figure 3.11 A potentially beautified EU premises access ceremony

3.8 Crowdsourcing for the Application of Beautiful Security

My second questionnaire aimed to determine whether applying my guidelines to ceremonies, or highlighting improvements based on my guidelines in existing ceremonies, had been effective. I wanted to find out whether the beautified alternative I present to the respondents is indeed considered a positive improvement, or not.

3.8.1 Design of the Second Questionnaire

I designed the questionnaire to present two alternative versions of the same ceremony: an original and a beautified one, the latter arrived at by applying one or more of the beautification guidelines. To that end, I presented respondents with the four different ceremonies I described in Section 3.7. Respondents were asked to answer by giving a preference on which of the two versions of each ceremony they think is the most beautiful. However, before proceeding, it is necessary to make some observations.

First, note that users are not specifically asked to express a preference on which ceremony they would prefer to use (this would have required an additional question for each ceremony presented asking for their preference), nor to respond that there was no difference between the two given ceremonies. Here I wanted to capture to what extent the users, with their understanding of security ceremonies, would have perceived the "beautified" versions as a positive, ultimately as a beautiful, improvement of the original ceremonies.

Second, statistical significance was not tested. This could be a part of a further investigation, considering also to carry out the investigation with a larger sample.

The questionnaire consisted of the following four closed-ended questions, allowing people to respond with a single preference:

Q1: You have to choose a new system to be authenticated to vote in an election. Which of the following two alternative systems is more beautiful?

1. A system that requires you to bring your "voting eligibility certificate" in addition to your ID Card.
2. A system that requires you to bring only an electronic ID document.

Q2: You have to choose a new system to log in on your laptop. Which of the following two alternative systems is more beautiful?

1. A system that requires you to type in your password.
2. A system that requires you to approach your laptop with your smartwatch.

Q3: You have to choose a new authentication system. Which of the following two alternative systems is more beautiful?

1. A system that requires you to insert a complex password (including requirements such as minimum number of characters, or the use of capital letters, numbers and special characters).
2. A system that requires you to insert whatever password you want but it limits the number of times you may type the password incorrectly before getting locked out.

Q4: The security officers at the entrance of a secure building have registered your ID and fingerprint on the system once you have completed the registration. Which of the following two alternatives to access the building from now on is more beautiful?

1. ID + fingerprint every time.
2. Fingerprint every time.

Q1 thus refers to the two versions of the Italian voting ceremony; Q2 to the two versions of the Laptop login ceremony; Q3 to the two versions of the Password setup ceremony; Q4 to the two versions of the EU premises access ceremony.

3.8.2 Findings

I administered the questionnaire to one hundred respondents using the CrowdFlower platform, not constraining respondents in terms of language or geography. As I remarked above, CrowdFlower workers span the globe, so heterogeneity of the sample is assumed. In order to obtain more focused answers from semi-expert respondents, I also administered the questionnaire to 24 students of the "Cryptography and Information Security" course that one of the authors is teaching. The students attending this module are computer science or mathematics students in the final year of their undergraduate studies or in the first year of their master's studies. We can thus call these respondents "security-savvy", in contrast to the "generic" respondents on CrowdFlower. Security-savvy respondents were asked the same questions Q1–Q4 but were also shown the MSCs of the ceremonies.

The combined results of the second questionnaire are shown in Figure 3.12 and I now wish to take stock and reflect on the responses that I received. The respondents provided me with extremely useful, and to some extent slightly surprising, feedback.

The answers to the first question clearly indicate that both generic and security-savvy users find the beautified Italian voting ceremony that I propose to be considerably more beautiful than the ceremony currently in use.

The answers to the fourth question indicate even more strongly that both generic and security-savvy users find the novel EU premises access ceremony that I propose to be considerably more beautiful than the ceremony currently in use.

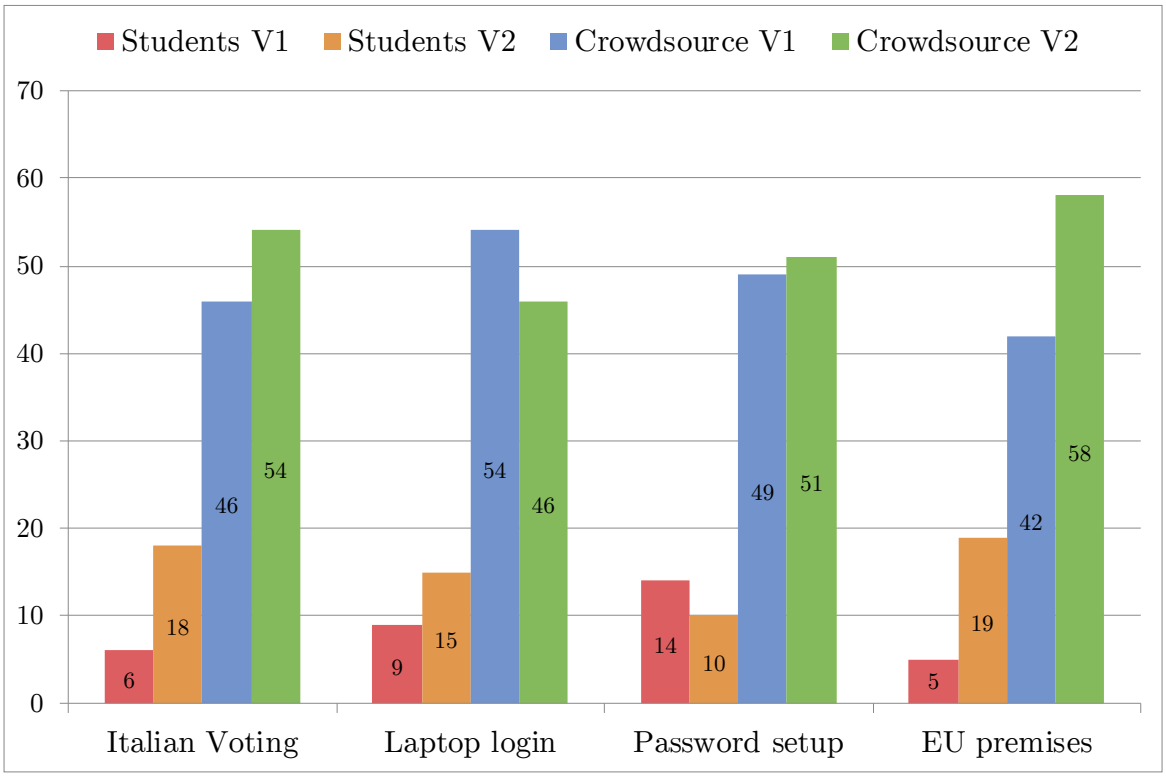


Figure 3.12 Data obtained from the second crowdsourcing

The answers to the second and the third question indicate that my respondents were almost equally split between the original versions of the second and third ceremonies I considered and their beautified versions. It is, however, instructive to analyse the responses in more detail.

Consider the password setup ceremony. It is interesting to note that the 100 generic respondents have a minimal preference for the proposed beautified ceremony that uses the new NIST rules (51 vs. 49), whereas almost 60% of the semi-expert students would rather engage in the original ceremony based on the 2003 NIST rules. I believe that these results are evidence to the fact that the respondents, especially the security-savvy ones, perceived the original ceremony to be more robust than the newly proposed ceremony, although the latter is potentially more lean and beautiful.

Finally, consider the laptop login ceremony. In this case, 54% of the generic respondents preferred the original ceremony, whereas 62.5% of the security-savvy respondents preferred the novel ceremony using the smartwatch. I believe that this indicates that young computer-science/mathematics students are more likely to embrace novel technologies than the cohort of generic respondents, which includes people of different age, education and background, who are thus more likely to meet such technologies with suspicion or even rejection (perhaps also due to the elitist nature of such technologies).

3.9 Beauty and Security: Friends or Foes

The beautification process allows the designer to reason on burdensome aspects of security ceremonies based on my guidelines. The considerations that the designer can make are user-based. However, the beautification process cannot be exempted from considering a security analysis of the ceremonies produced. Such a security analysis can, for instance, be carried out using formal methods as I have done in Chapter 5 for the Oyster and Single Sign-On ceremonies and in Chapter 6 for the ticket inspection ceremony, where I demonstrate how tools can support the formal analysis of security ceremonies in which human users play a key role. These analyses are typically carried out by security analysts rather than the ceremony designers.

What is interesting here is to focus on the designer's point of view, allowing him to carry out a first skimming of the security issues, which does not require a full-fledged formal analysis — which will be necessary at some point — but that provides the designer of a semi-formal state of mind to identify pitfalls and problems. For instance, one can reason if and how the beautification process has introduced errors and vulnerabilities that didn't exist before. The following are a couple of examples on how one can perform this type of reasoning.

Consider the Italian voting ceremony in its current version in Figure 3.2 and in its beautified version in Figure 3.4. The security property to be guaranteed is *authentication*, admitting to the voting phase the only voters that have been authenticated. Typical attacks for this property can be impersonification, but also man-in-the-middle attacks. Having highlighted the possible attacks, one can reason about two questions:

- Are these attacks possible in the beautified version of the ceremony?
- If, yes, were these attacks introduced by the beautification process?

Looking at the Italian voting ceremony in its beautified version, one can notice that a new channel has been introduced between the polling station employee and the system. On one hand, without any assumption on the security of the channel, this channel can be vulnerable to potential man-in-the-middle attacks, which can compromise the authentication security property. In fact, in this case, an attack can be performed in the beautified version and it has been introduced by the beautification process. On the other hand, in case the channel is secure using, for example VPNs over IPSec, then the channel is not vulnerable and the security is guaranteed in that part of the ceremony. So, if there is still a vulnerability of the authentication property, it has to be in another part of the ceremony, and since no other parts have been changed, the vulnerability is present also in the original ceremony. This means that the beautification process has not introduced a vulnerability.[4]

This kind of analysis can unveil these scenarios, which allow one to reason on the changes introduced to the beautification process. This type of analysis, however, does not aim to replace a formal analysis, but is complementary and helpful to the designer.

[4]More formally, one could prove the conditions under which attacks on the ceremony produced by the beautification process can be reduced to attacks to the original ceremony. Dually, one could identify the conditions under which a specific beautification process maintains security, meaning that if one has proved the security of the original ceremony then this specific beautification yields, provably, a secure beautified ceremony. These kinds of analysis are complex and require considerable work that goes beyond the context and scope of this thesis.

3.10 Conclusions

This chapter described how I went about finding a way to beautify security ceremonies, thereby making beautiful security achievable in practice. I first explored the literature to find out what beauty means to users, specifically in the context of interaction with software systems. I then explored general perceptions of security ceremonies by posing questions to crowdsourcing respondents. Having gained insights from this process, I proceeded to "beautify" four security ceremonies. I then posed another round of questions to crowdsourced participants to judge the success of my beautification of ceremonies. Overall, the outcome of the final stage confirmed that two of the four beautified ceremonies were indeed perceived to be more beautiful than they had previously been. The other two beautified ceremonies were not perceived to be more beautiful by the respondents.

The use of crowdsourcing enabled me to explore and implement the "beautiful" security paradigm, and gain feedback from a diverse and global audience. My experiments confirmed that security ceremony design has plenty of scopes to consider instilling beauty — with the general aim of achieving more beautiful security ceremonies. The assumption that people are normally attracted to beauty, as vastly substantiated by the relevant literature, will then strengthen such a design because it will be secure not just in isolation but *when used* by its users. Beauty may reinforce the designers' assumptions that the users will conform to the ceremony as intended.

A variety of real-world ceremonies would benefit from being subjected to a beautification process based on guidelines such as those proposed here. The process itself could be developed further, for example by combining and leveraging several guidelines at the same time. This will be particularly important in the case of ceremonies such as Password setup, on which my experiments showed that beauty will also need to convey robustness. The Laptop login ceremony, where newer technologies such as the Apple

Watch are used, might not yet be considered to be beautiful by the wider population today.

The methodology discussed here has reached the development stage of a proof of concept but I am aware that it must be developed further as this study is just skimming the surface of linking the perception of "beauty" with security. For instance, I have already repeated the crowdsourcing experiments over a larger sample population to reinforce the actual beautification guidelines and to find others. I am applying a qualitative methodology [140] on the data in order to classify the results. The next phase will be to understand if the potential new guidelines can be applied to the security ceremonies and in which instance I can confirm that the beatification has succeeded, taking advantage of another crowdsourcing or using different methods, e.g., focus groups.

Moreover, although beautifying existing ceremonies may sometimes lead to simplifying them, simplicity did not turn up in the categories that emerged from the analysis of answers to open question Q4. Arguably, simplifying a security ceremony might compromise security; as part of future works, I thus also plan to develop combinations of empirical, analytical and formal approaches [20, 131, 88, 104] to help ensure that when the new paradigm is applied to a ceremony, possibly simplifying it, it remains secure. This will require defining degrees of ceremony attractiveness for humans in a formal, logico-mathematical way, in order to then be able to reason about the interplay of beautification and security, ultimately ensuring that the beauty of a ceremony does not come at the expense of its security (which might be the case for some of the ceremonies I considered, e.g., if users choose weak passwords), but instead provably reinforces its security. I expect that this will ultimately lead to defining criteria that formalise when beautification preserves or reinforces security.

Chapter 4

What Are the Threats? (Charting the Threat Models of Security Ceremonies)

This chapter addresses the underlying, fundamental question of what are, and how to define, the threat models for a security ceremony. This work has been published as: Sempreboni, D., Bella, G., Giustolisi, R., and Viganò, L. (2019). What Are the Threats?(Charting the Threat Models of Security Ceremonies). In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE - Volume 2: SECRYPT*, pages 161–172. Scitepress Digital Library. [147]

4.1 Motivation

As mentioned in Chapter 1, there is increasing awareness that security protocols need better attention to their "human element" than what is traditionally paid. This trend is confirmed by recent works, both on the formal [20, 100, 16] and on the practical level [74, 23, 88, 156, 110, 158].

While existing works (such as the ones cited above) define threats that are reasonable, they generally fail to treat the threats *systematically* within the given ceremony, hence potentially missing relevant combinations of threats. For example, a vulnerability in a website might be exploited by a specific sequence of user actions, which an attacking third party would need to deceive the user to take. This attack cannot be discussed without admitting a complicated threat model that combines *at the same time* (but without any form of collusion): *(i)* a bug in the website, *(ii)* a user who makes *wrong* choices and *(iii)* an active attacker capable of deception.

A huge variety of similar situations may underlie modern security ceremonies, and that variety is, in turn, due to the variety of the ceremonies themselves, with different levels of intricacies and innumerable applications, ranging from pre-purchasing a cinema ticket via the web to obtaining an extended validation certificate. A remarkable, recent and large-scale, attack saw the "Norbertvdberg" hacker advertise his online seed generator iotaseed.io through Google for a semester; but the generator was bogus, so that Norbertvdberg could hack a number of seeds and harvest a total of $3.94 million worth of IOTA at the only extra effort of mounting a DDoS against the IOTA network to prevent investigation. Here, both the hacker and his website acted maliciously, though arguably in different ways, mounting a complex socio-technical attack against both users and the IOTA infrastructure [80].

Therefore, it is clear that security ceremonies don't succumb to the "one threat model to rule them all" proviso as security protocols traditionally did with the Dolev-Yao attacker model [56]. In fact, the Dolev-Yao model has proved to be very successful for the analysis of security protocols, where the almighty attacker "rules" over the other protocol agents who are assumed to behave only as prescribed by the protocol specification. However, in the case of security ceremonies such an attacker provides an inherent "flattening" that likely makes one miss relevant threat scenarios. By analogy,

one could say that the Dolev-Yao attacker is a powerful hammer... but to a man with a hammer, everything looks like a nail, forgetting that there are also screws and nuts and bots (for which a hammer is inadequate). I advocate that for security ceremonies we need an approach that provides a birds-eye view, an "over-view" that allows one to consider what are the different threats and where they lie, with the ultimate aim of finding novel attacks.

4.2 Contributions

The main contribution of this chapter is thus *the systematic definition of an encompassing method to build the full threat model chart for security ceremonies from which one can conveniently reify the threat models of interest for the ceremony under consideration.* More specifically, I provide four concrete contributions.

1. Classification and labelling The method starts with a classification of the principals participating in security ceremonies and continues with a motivated labelling system for their actions and principals. Contrarily to some of the mentioned works, my method abstracts away from the reasons that determine human actions such as error.

2. Combination of the principal labels The method systematically combines the principal labels to derive a number of threat models that, together, form the full chart of threat models for the ceremony. We shall see that the higher the number of principals in a ceremony, the more complicated its full threat model chart: we shall represent it as a table, where each line signifies a specific threat model.

3. Testing my method For concreteness, I demonstrate the application of the method on four ceremonies that have already been considered, albeit at different levels of detail and analysis, in the literature:

- MP-Auth [16, 99],
- Opera Mini [131],
- Cash-point [133], and
- the Danish Mobilpendlerkort ceremony [68].

I discuss how the full threat model chart suggests some interesting threats that haven't been investigated although they are well worth of scrutiny. In particular, it turns out that the Danish Mobilpendlerkort ceremony is vulnerable to the combination of an attacking third party and a malicious phone of the ticket holder's. The threat model that leads to this vulnerability has not been considered so far and arises here thanks to my charting method.

4. Formal proof using model checking To demonstrate the relevance of the chart, I modelled and analysed this threat model using the formal and automated tool Tamarin [161], which enables the unbounded verification of security protocols, although it is important to highlight that my method is generic and can be used with any tool for the analysis of security protocols and ceremonies.

4.3 Outline

A full threat model chart for security ceremonies is in Section 4.4. The application of my chart on three example is in Section 4.5 and Section 4.6: MP-Auth, Opera-Mini and Cash-point are discussed in Section 4.5, whereas I devote the whole of Section 4.6 to the Danish Mobilpendlerkort ceremony. Conclusions are discussed in Section 4.7.

4.4 Charting the Threat Models of Security Ceremonies

I designed the threat models considering three components: the *principals*, the *actions* that each principal performs and *the combination of principals and actions* resulting in *scenarios*. The charting that results from these three components will be a meta-model to help security analysts to consider all the resulting threats within a security ceremony.

4.4.1 Principals

The principals that may participate in a security ceremony C are

- a number of technical systems,
- a number of humans and
- an attacking third party.

A technical system $TechSystem_i$ is one that can be programmed and works by executing its program (using the terminology of security protocol analysis, these are *honest* principals that behave according to the protocol specification). Depending on the architectural level represented in the given ceremony, a technical system could be either a piece of hardware, say a network node, or of software, say the program executing on that node. I envisage zero or more technical systems participating in the ceremony.

A human principal $Human_i$ may variously interact with other humans or with the technical systems. For example, $Human_1$ may interact directly with $Human_2$ in a face-to-face ceremony without technology, or respectively with technical systems $TechSystem_1$ and $TechSystem_2$, which may in turn interact between themselves through a security protocol. As with technical systems, zero or more humans could participate

in the ceremony, and the case of zero humans would take us back to the realm of security protocols.

I also consider an *attacking third party ATP*, which can be seen as a combination of colluding attackers, in the style of the Dolev-Yao attacker. However, the *ATP* is inherently *socio-technical*, namely it is capable of interacting both with a human by engaging in human actions, and with a technical system by engaging in digital communication with it. The *ATP* could thus be interpreted as the union of some $Human_x$ and some $TechSystem_y$; the capabilities of the *ATP* will become clearer below.

More formally, the principals of a ceremony C can be formalised as follows:

$$\begin{aligned}
Technicals(\mathsf{C}) &:= \{TechSystem_i \mid i = 1, \ldots, n\} \\
Humans(\mathsf{C}) &:= \{Human_j \mid j = 1, \ldots, m\} \\
ATP(\mathsf{C}) &:= \{ATP_k \mid k \leq 1\} \\
Principals(\mathsf{C}) &:= Technicals(\mathsf{C}) \cup \\
&\quad\ Humans(\mathsf{C}) \cup \\
&\quad\ ATP(\mathsf{C})
\end{aligned}$$

where, as is standard, I assume that if $n = 0$, then C doesn't contain any technical system principal. Similarly, if $m = 0$, then C doesn't contain any human principal, and if $k = 0$, then there is no attacking third party.

Let's consider the Italian Voting ceremony C in Section 3.7.1. The principals of this ceremony are:

$$Principals(\mathsf{C}) := \quad Humans(\mathsf{C}) \quad := \quad \{user, polling\ station\ employee\}$$

whereas, if I consider its "beautified" version C', the principals are:

$$
\begin{aligned}
\mathit{Technicals}(\mathsf{C}') &:= \{\mathit{system}\} \\
\mathit{Humans}(\mathsf{C}') &:= \{\mathit{user}, \mathit{polling\ station\ employee}\} \\
\mathit{Principals}(\mathsf{C}') &:= \mathit{Technicals}(\mathsf{C}') \cup \\
&\quad\ \mathit{Humans}(\mathsf{C}') \cup
\end{aligned}
$$

4.4.2 Information and Actions

Principals of a security ceremony exchange *information* of various types, such as a password that a human may type, a ticket on a smartphone that a human may show to an inspector, a PDF that a technical system may display to a human, or a binary message, which is typically exchanged between technical systems.

More formally, I express the heterogeneity of the information exchanged in the given ceremony C by introducing a free type

$$\mathit{Information}(\mathsf{C})\,.$$

I don't need to specify this type in more detail here, but I take it to cover objects being used and exchanged in the ceremony (cards, PDFs, etc.) as well as data being transmitted (URLs, binary messages, etc.). Of course, in some cases, $\mathit{Information}(\mathsf{C})$ may reduce to the standard type of abstract encrypted messages typically used in security protocol analysis.

A ceremony C comes with a predefined set of *actions* to be performed by a principal with another one, or many more principals, through the exchange of some information. Actions include the sending of messages by technical systems, but also commands that humans may give to technical systems.

The most common sets of technical, human and attacking third party actions can be defined as ternary relations $\mathcal{A}_{\mathsf{C},TS}$, $\mathcal{A}_{\mathsf{C},H}$ and $\mathcal{A}_{\mathsf{C},ATP}$, respectively:

$$\begin{aligned}
&\mathit{TechnicalActions}(\mathsf{C}) := \\
&\qquad \mathcal{A}_{\mathsf{C},TS}(\mathit{Technicals}, \mathit{Information}, \mathit{Principals}) \\
&\mathit{HumanActions}(\mathsf{C}) := \\
&\qquad \mathcal{A}_{\mathsf{C},H}(\mathit{Humans}, \mathit{Information}, \mathit{Principals}) \\
&\mathit{ATPActions}(\mathsf{C}) := \\
&\qquad \mathcal{A}_{\mathsf{C},ATP}(\{\mathit{ATP}\}, \mathit{Information}, \mathit{Principals})
\end{aligned}$$

Actions can be projected onto given principals $\mathit{TechSystem}_i$ or Human_j:

$$\begin{aligned}
&\mathit{TechnicalActions}(\mathsf{C})|_{\mathit{TechSystem}_i} := \\
&\quad \mathcal{A}_{\mathsf{C},TS}(\{\mathit{TechSystem}_i\}, \mathit{Information}, \mathit{Principals}) \\
&\mathit{HumanActions}(\mathsf{C})|_{\mathit{Human}_j} := \\
&\qquad \mathcal{A}_{\mathsf{C},H}(\{\mathit{Human}_j\}, \mathit{Information}, \mathit{Principals})
\end{aligned}$$

The relations may, however, also be binary (e.g., a principal broadcasting some information to all other principals, who thus don't need to be specified as the third parameter) or quaternary (e.g., a principal sending a message to a principal through another principal). So, in general, I define the set of actions of a ceremony as the union of the set of actions of its principals:

$$\begin{aligned}
\mathit{Actions}(\mathsf{C}) \;:=\; & \mathit{TechnicalActions}(\mathsf{C}) \cup \\
& \mathit{HumanActions}(\mathsf{C}) \cup \\
& \mathit{ATPActions}(\mathsf{C})
\end{aligned}$$

4.4.3 Action Labels

A key step of my method is the *labelling of actions* and the *labelling of principals*. I introduce the former in this subsection, and the latter in the next subsection.

Each action of a $TechSystem_i$ of a given ceremony C can be labelled as follows:

- *normal*, to indicate actions that are prescribed by the ceremony, analysing a received message and generating another one to send out;
- *bug*, to indicate an unwanted technical deviation from *normal*, hence occurring without a specific goal and normally unexpectedly, such as Heartbleed or Shellshock;
- *malicious*, to indicate a deliberate technical deviation from *normal*, hence implemented with the deliberate goal of someone's profit. Examples of malicious actions are the execution of malware, such as a backdoor or a trojan, which the producer inserted at production time, in which case the profit would be the producer's, or the execution of post-exploitation code injected by the *ATP* while the technical system is in use, in which case the profit would be the *ATP*'s.

The second and third labels could be usefully equipped with parameters carrying out the relevant details of the bug or of the malicious action. More formally, for each ceremony, the following function can be defined:

$$technical\_action\_label_{\mathsf{C}} : \\ TechnicalActions(\mathsf{C}) \rightarrow \{normal, bug, malicious\}$$

Each action of a $Human_j$ of a given ceremony C can be labelled as follows:

- *normal*, to indicate actions that are prescribed by the given ceremony, such as opening a website, launching an app or handing a paper receipt over to another human;

- *error*, to indicate an unwanted human deviation from *normal*, hence occurring without a specific goal and normally unexpectedly, such as typing a *wrong* (i.e., different from the one the human wanted) password, making the wrong choice or handing out a wrong item;
- *choice*, to indicate a deliberate human deviation from *normal*, hence occurring with the deliberate goal of someone's profit. Example choice actions include those just given as errors, provided that they are reinterpreted towards profit. Other example choices include attempts to attack a technical system, for example by exploiting a vulnerability. Profit could be personal for the human making the choice or it could be *ATP*'s profit if the human choice is due to deception.

Also in this case, the second and third labels could be usefully equipped with parameters carrying out the relevant details of the error or of the choice. More formally, for each ceremony, the following function can be defined:

$$human\_action\_label_{\mathsf{C}} : \\ HumanActions(\mathsf{C}) \rightarrow \{normal, error, choice\}$$

Finally, the *ATP* either does nothing (i.e., is not present at all) or attacks the ceremony. To do the latter, it may engage in the ceremony by performing technical actions that are malicious or human actions of choice:

$$atp\_action\_label_{\mathsf{C}} : \\ ATPActions(\mathsf{C}) \rightarrow \{malicious, choice\}$$

This is where one can see the difference between my *ATP* and the Dolev-Yao attacker. The latter controls the network and can impersonate other principals by acting honestly in a protocol run (but cannot break cryptography, yet computational extensions have been proposed). By contrast, my *ATP* is an external principal and cannot be an insider

of the ceremony — the attacks that can happen in that case are covered in my model by appropriately labelled actions of the humans or technical systems participating in the ceremony (in particular, labelled as choice, error, bug or malicious). Similarly, an attacker that simply participates honestly in a ceremony can be expressed by humans and technical systems executing the ceremony with an "empty" external attacking third party. Collusion between an insider and an outsider can also be modelled that way.

The charting represents a meta-model based on "intentions" of principals. In particular, the ATP is a principal with dishonest intentions. It is not meant to be a direct substitute of the Dolev-Yao attacker as it is not at the same level of granularity. In fact, the ATP in the chart encompasses the Dolev-Yao attacker as one of the possible cases (this special case can be captured by appropriately labelling the actions and the principals of the ceremony, as I shall clarify below) and, indeed, in the Single Sign-On ceremony in Chapter 5 the attacker used is the Dolev-Yao attacker as it covers the "intentions" expressed by the chart. On the other hand, what the charting allows us to do that we were not able to do before is, in fact, to reason without the Dolev-Yao attacker as shown in the Danish Mobilpendlerkort ceremony considered later in this chapter. In short, the charting provides a birds-eye view on which threat model can be used in the different ceremonies considered, as illustrated, for example, by the Oyster ceremony and Single Sign-On ceremony considered in Chapter 5 where I introduce scenarios with and without the Dolev-Yao attacker.

4.4.4 Scenario Labels and Principal Labels

We have just seen how to label each action that the principals of a given ceremony C may execute. The labelling system can be lifted at the level of principals. For example, a principal is labelled as

- *normal* if the principal is a technical system or a human and all actions of the principal are labelled as *normal*; or as
- *choice* if the principal is a human and all actions of the principal are either labelled as *choice* or as *normal*; and so on.

More precisely, I identify 11 relevant groups of action labels that a principal, (depending on whether it is a technical system, a human or the ATP) may use, and define them as a set of scenarios:

$$
\begin{aligned}
\mathit{scenario}_1(p) &:= p \in \mathit{Technicals}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{TechnicalActions}(\mathsf{C}, p) \text{ it is } a = \mathit{normal} \\
\mathit{scenario}_2(p) &:= p \in \mathit{Humans}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{HumanActions}(\mathsf{C}, p) \text{ it is } a = \mathit{normal} \\
\mathit{scenario}_3(p) &:= p \in \mathit{Technicals}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{TechnicalActions}(\mathsf{C}, p) \\
&\quad \text{it is } a = \mathit{bug} \text{ or } a = \mathit{normal} \\
\mathit{scenario}_4(p) &:= p \in \mathit{Technicals}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{TechnicalActions}(\mathsf{C}, p) \\
&\quad \text{it is } a = \mathit{malicious} \text{ or } a = \mathit{normal} \\
\mathit{scenario}_5(p) &:= p = \mathit{ATP}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{ATPActions}(\mathsf{C}, p) \text{ it is } a = \mathit{malicious} \\
\mathit{scenario}_6(p) &:= p \in \mathit{Humans}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{HumanActions}(\mathsf{C}, p) \\
&\quad \text{it is } a = \mathit{error} \text{ or } a = \mathit{normal} \\
\mathit{scenario}_7(p) &:= p \in \mathit{Humans}(\mathsf{C}) \text{ and} \\
&\quad \forall a \in \mathit{HumanActions}(\mathsf{C}, p) \\
&\quad \text{it is } a = \mathit{choice} \text{ or } a = \mathit{normal}
\end{aligned}
$$

$$
\begin{aligned}
scenario_8(p) &:= p = ATP(\mathsf{C}) \text{ and} \\
&\quad \forall a \in ATPActions(\mathsf{C}, p) \text{ it is } a = choice \\
scenario_9(p) &:= p \in Technicals(\mathsf{C}) \text{ and} \\
&\quad \forall a \in TechnicalActions(\mathsf{C}, p) \\
&\quad \text{it is } a = bug \text{ or } a = malicious \text{ or } a = normal \\
scenario_{10}(p) &:= p \in Humans(\mathsf{C}) \text{ and} \\
&\quad \forall a \in HumanActions(\mathsf{C}, p) \\
&\quad \text{it is } a = error \text{ or } a = choice \text{ or } a = normal \\
scenario_{11}(p) &:= p = ATP(\mathsf{C}) \text{ and} \\
&\quad \forall a \in ATPActions(\mathsf{C}, p) \\
&\quad \text{it is } a = malicious \text{ or } a = choice
\end{aligned}
$$

This list of scenarios emphasises all combinations of labels that I derive systematically and deem significant for the principals, namely *bug* + *malicious* for technical systems, *error* + *choice* for humans and *malicious* + *choice* for the attacking third party. The scenarios show that only technical systems and humans can take *normal* actions, while only technical systems and the attacking third party can take *malicious* actions, and only humans and the attacking third party can take *choice* actions. The scenarios also confirm that the attacking third party is the only principal who can combine both *malicious* and *choice* actions.

In order to express the scenarios a principal may follow, I introduce a function that maps principals to sets of scenarios, intuitively yielding, for a given principal, the set of scenarios that the principal may follow:

$$scena_{\mathsf{C}} : Principals(\mathsf{C}) \to 2^{\{scenario()_k | k=1,\ldots,11\}}$$

with the obvious constraints that:

- if $p \in \mathit{Technicals}(\mathsf{C})$ then $\mathit{scena}_{\mathsf{C}}(p) \in 2^{\{\mathit{scenario}_1(p), \mathit{scenario}_3(p), \mathit{scenario}_4(p), \mathit{scenario}_9(p)\}}$
- if $p \in \mathit{Humans}(\mathsf{C})$ then $\mathit{scena}_{\mathsf{C}}(p) \in 2^{\{\mathit{scenario}_2(p), \mathit{scenario}_6(p), \mathit{scenario}_7(p), \mathit{scenario}_{10}(p)\}}$
- if $p = \mathit{ATP}$ then $\mathit{scena}_{\mathsf{C}}(p) \in 2^{\{\mathit{scenario}_5(p), \mathit{scenario}_8(p), \mathit{scenario}_{11}(p)\}}$

This function is in general total because every principal will be associated to some scenarios but not surjective; hence, some (sets of) scenarios may not have any principal that is mapped into them in the given ceremony C. Intuitively, this means that no principal acts according to those scenarios in the given ceremony. I then introduce the scenario label function:

$$\begin{aligned} \mathit{scenario\_label}_{\mathsf{C}} : \{\mathit{scenario}_k() \mid k = 1, \ldots, 11\} \rightarrow \\ \{\mathit{normal}, \mathit{bug}, \mathit{malicious}, \mathit{error}, \mathit{choice}, \\ \mathit{bug} + \mathit{malicious}, \mathit{error} + \mathit{choice}, \\ \mathit{malicious} + \mathit{choice}\} \end{aligned}$$

This function is in general partial, which means that C may not allow us to define the principal label on some scenarios, precisely on those that are not associated to any principal by the function $\mathit{scena}_{\mathsf{C}}$. For a given scenario s, the function is defined as:

$$
\begin{aligned}
&scenario\_label_{\mathsf{C}}(s) = \\
&\quad \begin{cases}
normal & \text{if } s = scenario_1() \text{ or} \\
 & \quad\ s = scenario_2() \\
bug & \text{if } s = scenario_3() \\
malicious & \text{if } s = scenario_4() \text{ or} \\
 & \quad\ s = scenario_5() \\
error & \text{if } s = scenario_6() \\
choice & \text{if } s = scenario_7() \text{ or} \\
 & \quad\ s = scenario_8() \\
bug + malicious & \text{if } s = scenario_9() \\
error + choice & \text{if } s = scenario_{10}() \\
malicious + choice & \text{if } s = scenario_{11}()
\end{cases}
\end{aligned}
$$

The labels of a principal can be defined as the labels of the scenarios the principal may follow according to function $scena_{\mathsf{C}}(p)$. More formally:

$$
\begin{aligned}
&PLTechnicals_{\mathsf{C}}(p) := \\
&\quad \{l \mid l = scenario\_label_{\mathsf{C}}(s) \text{ for } s \in scena_{\mathsf{C}}(p)\} \\
&PLHumans_{\mathsf{C}}(p) := \\
&\quad \{l \mid l = scenario\_label_{\mathsf{C}}(s) \text{ for } s \in scena_{\mathsf{C}}(p)\} \\
&PLATP_{\mathsf{C}} := \\
&\quad \{l \mid l = scenario\_label_{\mathsf{C}}(s) \text{ for } s \in scena_{\mathsf{C}}(p)\}
\end{aligned}
$$

Depending on the actions the various principals may take in C, it is clear from the above definitions that:

$$\begin{aligned} n(PLTechnicals_{\mathsf{C}}(p)) &\leq 4 \\ n(PLHumans_{\mathsf{C}}(p)) &\leq 4 \\ n(PLATP_{\mathsf{C}}) &\leq 3 \end{aligned}$$

4.4.5 Building the Full Threat Model Chart

As mentioned in Section 2.2, a *security ceremony* expands a security protocol with everything that is considered out-of-band to it, including, in particular, the mistakes that human users might make when participating actively in the security ceremony. The Full Threat Model Chart has been built systematically taking into account the important aspects that characterize the security ceremonies (e.g., who is participating such as technical systems, human participants, external attackers but also communication channels between human participants and communication channels between technical systems and human participants).

The chart, in fact, considers the three types of principals *Technicals*, *Humans* and *ATP*. Furthermore, each of the principals comes with a set of actions that represents the intentions of a specific principal in the security ceremony. Because of its generality, the chart is able to capture principals' "intentions" that span from social engineering to guessing situations. At the end, the full chart of threat models for a given ceremony C can be built by systematically combining the labels of all principals in $Principals(\mathsf{C})$, namely by building the set $PLTechnicals_{\mathsf{C}}(p)$ for each technical system p, the set $PLHumans_{\mathsf{C}}(p)$ for each human p and the set $PLATP_{\mathsf{C}}$ and, finally, by composing their elements exhaustively. For example, suppose that C features a human principal and two technical systems that, respectively, use all possible action labels, while no attacking third party is assumed to exist. It means that I have to build three sets of

principal labels:

$$\begin{aligned} &PLHumans_{\mathsf{C}}(Human) = \\ &\qquad \{normal, error, choice, error + choice\} \\ &PLTechnicals_{\mathsf{C}}(TechSystem_1) = \\ &\qquad \{normal, bug, malicious, bug + malicious\} \\ &PLTechnicals_{\mathsf{C}}(TechSystem_2) = \\ &\qquad \{normal, bug, malicious, bug + malicious\} \end{aligned}$$

Due to the cardinality of such sets, the full threat model chart for C has width $n(Principals(\mathsf{C})) = 3$ and depth:

$$\begin{aligned} n(PLTechnicals_{\mathsf{C}}(Human)) \quad &\times \\ n(PLHumans_{\mathsf{C}}(TechSystem_1)) \quad &\times \\ n(PLHumans_{\mathsf{C}}(TechSystem_2)) \quad &= \quad 4^3 = 64. \end{aligned}$$

It is given in Table 4.1.

As another example, consider a ceremony C' that extends C with an attacking third party that only interferes with the human principals in all scenarios, but not with the technical systems. It means that:

$$PLATP_{\mathsf{C}'} = \{choice\}.$$

As a consequence, the full threat model chart for C' doubles (the height of) the chart in Table 4.1, with the new half being the same as the first half but with an extra column for the attacking third party repeating *choice* for 64 times.

#	*Human*	$TechSystem_1$	$TechSystem_2$
1	*normal*	*normal*	*normal*
2	*normal*	*normal*	*bug*
3	*normal*	*normal*	*malicious*
4	*normal*	*normal*	*bug+malicious*
5	*normal*	*bug*	*normal*
6	*normal*	*bug*	*bug*
7	*normal*	*bug*	*malicious*
8	*normal*	*bug*	*bug+malicious*
9	*normal*	*malicious*	*normal*
10	*normal*	*malicious*	*bug*
11	*normal*	*malicious*	*malicious*
12	*normal*	*malicious*	*bug+malicious*
13	*normal*	*bug+malicious*	*normal*
14	*normal*	*bug+malicious*	*bug*
15	*normal*	*bug+malicious*	*malicious*
16	*normal*	*bug+malicious*	*bug+malicious*
17	*error*	*normal*	*normal*
18	*error*	*normal*	*bug*
19	*error*	*normal*	*malicious*
20	*error*	*normal*	*bug+malicious*
21	*error*	*bug*	*normal*
22	*error*	*bug*	*bug*
23	*error*	*bug*	*malicious*
24	*error*	*bug*	*bug+malicious*
25	*error*	*malicious*	*normal*
26	*error*	*malicious*	*bug*
27	*error*	*malicious*	*malicious*
28	*error*	*malicious*	*bug+malicious*
29	*error*	*bug+malicious*	*normal*
30	*error*	*bug+malicious*	*bug*
31	*error*	*bug+malicious*	*malicious*
32	*error*	*bug+malicious*	*bug+malicious*

Table 4.1 The full threat model chart for a ceremony with a human principal, two technical systems and no attacking third party

#	*Human*	$TechSystem_1$	$TechSystem_2$
33	*choice*	*normal*	*normal*
34	*choice*	*normal*	*bug*
35	*choice*	*normal*	*malicious*
36	*choice*	*normal*	*bug+malicious*
37	*choice*	*bug*	*normal*
38	*choice*	*bug*	*bug*
39	*choice*	*bug*	*malicious*
40	*choice*	*bug*	*bug+malicious*
41	*choice*	*malicious*	*normal*
42	*choice*	*malicious*	*bug*
43	*choice*	*malicious*	*malicious*
44	*choice*	*malicious*	*bug+malicious*
45	*choice*	*bug+malicious*	*normal*
46	*choice*	*bug+malicious*	*bug*
47	*choice*	*bug+malicious*	*malicious*
48	*choice*	*bug+malicious*	*bug+malicious*
49	*error+choice*	*normal*	*normal*
50	*error+choice*	*normal*	*bug*
51	*error+choice*	*normal*	*malicious*
52	*error+choice*	*normal*	*bug+malicious*
53	*error+choice*	*bug*	*normal*
54	*error+choice*	*bug*	*bug*
55	*error+choice*	*bug*	*malicious*
56	*error+choice*	*bug*	*bug+malicious*
57	*error+choice*	*malicious*	*normal*
58	*error+choice*	*malicious*	*bug*
59	*error+choice*	*malicious*	*malicious*
60	*error+choice*	*malicious*	*bug+malicious*
61	*error+choice*	*bug+malicious*	*normal*
62	*error+choice*	*bug+malicious*	*bug*
63	*error+choice*	*bug+malicious*	*malicious*
64	*error+choice*	*bug+malicious*	*bug+malicious*

Table 4.1 (Continued) The full threat model chart for a ceremony with a human principal, two technical systems and no attacking third party

4.5 Using the Full Threat Model Chart

In this section, I show how my method for the definition of a full threat model chart can be usefully applied to existing ceremonies. Depending on the number of principals and the scenarios they follow, my aim is to generate a chart in the style of that in Table 4.1 for each ceremony, and then use the chart to identify the relevant threat models for the ceremony.

I discuss three example security ceremonies that have already been analysed to some extent in the literature: MP-Auth [16, 99], Opera Mini [131] and Cash-point [133]. Once their respective full threat model chart is available, it can be used to address which rows, namely which threat models have already been investigated. Additionally, the chart can be used to pinpoint other relevant threat models that I suggest to consider for further scrutiny of the ceremony, for example by formal analysis.

4.5.1 MP-Auth

The MP-Auth ceremony [16, 99] authenticates a human to a server using the human's platform and his personal device, to which the human has exclusive access. The main idea of the ceremony is that the human never needs to enter his password on the platform because this may be controlled by an attacker. The device has the public key of the server preinstalled. In short, the ceremony proceeds as follows: the human enters the name of the server he wants to communicate with on the platform, which then initiates communication with the server. The server in turn communicates with the device through the platform. The device displays the identity of the server to the human, who checks if this corresponds to the server he wants to communicate with. If it does, then he enters his password and identity on the device. Then, the device sends the login information to the platform, which relays it to the server.

According to my method, this ceremony encompasses one human principal and three technical systems. Precisely:

$$\begin{aligned} \mathit{Humans}(\mathit{MP\text{-}Auth}) &:= \{\mathit{Human}\} \\ \mathit{Technicals}(\mathit{MP\text{-}Auth}) &:= \\ &\quad \{\mathit{Platform}, \mathit{Device}, \mathit{Server}\} \end{aligned}$$

I also allow for an attacking third party, so that the resulting principals are:

$$\begin{aligned} \mathit{Principals}(\mathit{MP\text{-}Auth}) := \quad & \mathit{Humans}(\mathit{MP\text{-}Auth}) \cup \\ & \mathit{Technicals}(\mathit{MP\text{-}Auth}) \cup \\ & \mathit{ATP}(\mathit{MP\text{-}Auth}) \end{aligned}$$

The most general case in which every principal gets all possible labels sees:

$$\begin{aligned} n(\mathit{PLHumans}_{\mathit{MP\text{-}Auth}}(\mathit{Human})) &= 4 \\ n(\mathit{PLTechnicals}_{\mathit{MP\text{-}Auth}}(\mathit{Platform})) &= 4 \\ n(\mathit{PLTechnicals}_{\mathit{MP\text{-}Auth}}(\mathit{Device})) &= 4 \\ n(\mathit{PLTechnicals}_{\mathit{MP\text{-}Auth}}(\mathit{Server})) &= 4 \\ n(\mathit{PLATP}_{\mathit{MP\text{-}Auth}}) &= 2 \end{aligned}$$

Hence, the full threat model chart has $4^4 \cdot 2^1 = 512$ lines. The threat models considered in previous work [16] can be reinterpreted in my chart as shown in Table 4.2.

	Human	*Platform*	*Device*	*Server*	*ATP*
(a)	*error*	*normal*	*normal*	*normal*	*malicious*
(b)	*choice*	*normal*	*normal*	*normal*	*malicious*

Table 4.2 The two threat models already considered for the MP-Auth ceremony

However, my chart also highlights at least three more relevant threat models that haven't been considered yet. These are summarised in Table 4.3.

	Human	*Platform*	*Device*	*Server*	*ATP*
(c)	*error*	*bug*	*normal*	*normal*	*malicious*
(d)	*error*	*normal*	*bug*	*normal*	*malicious*
(e)	*choice*	*malicious*	*normal*	*normal*	*malicious*

Table 4.3 Additional threat models relevant for the MP-Auth ceremony.

Threat model (c) considers a human that makes errors while interacting with a buggy platform under the attack of an *ATP*; (d) considers the interaction with a buggy device; (e) considers a malicious platform and a malicious *ATP* as well as a human agent who decides to misbehave. It is clear that my chart helped distil out relevant threat models that have been neglected so far, and that a formal analysis is required to detect potential vulnerabilities entailed by the new threat models. In fact, differently from (a) and (b) in Table 4.2, the new cases in Table 4.3 add potential weak points that might lead to new attacks.

4.5.2 Opera Mini

The Opera Mini ceremony [131] begins with the user of a smartphone typing the address of his bank's website into the Opera Mini web browser. HTTPS connections are opened between the smartphone and Opera's Server, and between Opera's server and the bank's server. The request for the page is then passed through to the bank, which replies with its customer login page. The Opera server renders this page and sends the compressed output to the user's smartphone device. On the smartphone, Opera Mini then displays the webpage, including the padlock symbol. The user sees the padlock symbol and, if he chooses to input his login information and password,

then this is sent back to the bank's server via the Opera encrypted channel, decrypted at the Opera Server, and then re-encrypted and sent on to the bank's server via the HTTPS channel.

I don't allow for an attacking third party here, so, according to my method, the principals of the ceremony are:

$$
\begin{aligned}
&\mathit{Humans}(\mathit{Opera\text{-}Mini}) := \{\mathit{Human}\}\\
&\mathit{Technicals}(\mathit{Opera\text{-}Mini}) :=\\
&\qquad\qquad \{\mathit{Device}, \mathit{Opera\text{-}Server}, \mathit{Bank\text{-}Server}\}\\
&\mathit{Principals}(\mathit{Opera\text{-}Mini}) :=\\
&\quad \mathit{Humans}(\mathit{Opera\text{-}Mini}) \ \cup \ \mathit{Technicals}(\mathit{Opera\text{-}Mini})
\end{aligned}
$$

The most general case in which every principal gets all possible labels sees:

$$
\begin{aligned}
n(\mathit{PLHumans}_{\mathit{Opera\text{-}Mini}}(\mathit{Human})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Opera\text{-}Mini}}(\mathit{Device})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Opera\text{-}Mini}}(\mathit{Opera\text{-}Server})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Opera\text{-}Mini}}(\mathit{Bank\text{-}Server})) &= 4
\end{aligned}
$$

Hence, the full threat model chart has $4^4 = 256$ lines. The only threat model considered in [131] can be reinterpreted in my chart as shown in Table 4.4.

	Human	*Device*	*Opera-Server*	*Bank-Server*
(a)	*normal*	*bug*	*normal*	*normal*

Table 4.4 The threat model already considered for the Opera Mini ceremony.

However, my chart also highlights at least three more relevant threat models that haven't been considered yet. These are summarised in Table 4.5.

	Human	*Device*	*Opera-Server*	*Bank-Server*
(b)	*error*	*normal*	*normal*	*normal*
(c)	*error*	*normal*	*normal*	*malicious*
(d)	*choice*	*normal*	*normal*	*normal*

Table 4.5 Additional threat models for the Opera Mini ceremony.

The threat model (a) analysed in [131] considers just a buggy platform. I believe that it would be interesting to consider also threat model (b), where the human simply makes errors using the Opera Mini browser possibly, threat model (c), under the presence of a malicious *Bank-Server*. Threat model (d), instead, presumes that the 3 technical systems don't deviate from their specification but the human deliberately does in order to seek an advantage.

4.5.3 Cash-point

The *cash-point ceremony* [133] supports a basic ATM service, providing balance information and cash. Once a card is inserted, the machine checks its validity. If the card is not valid, then it is retained and an appropriate message is displayed. If the card is valid, then the machine asks the user to enter a PIN. Once this is done, the user can select one of the two options: withdraw cash or check balance. If the balance option is selected, then the machine releases the card and, once the card has been removed and after some delay, prints a receipt with the balance information. If the withdraw cash option is selected, then the user is asked to select the desired amount and (if the amount is within the limits), after some delay, the machine releases the card and, once it has been removed, outputs the cash.

According to my method, the principals of the ceremony are:

$$Humans(Cash\text{-}Point) := \{Human\}$$
$$Technicals(Cash\text{-}Point) := \{ATM\}$$

I also allow for an attacking third party, so that the resulting principals are:

$$\begin{aligned} Principals(Cash\text{-}Point) := \quad & Humans(Cash\text{-}Point) \cup \\ & Technicals(Cash\text{-}Point) \cup \\ & ATP(Cash\text{-}Point) \end{aligned}$$

The most general case in which every principal gets all possible labels sees:

$$\begin{aligned} n(PLHumans_{MP\text{-}Auth}(Human)) &= 4 \\ n(PLTechnicals_{MP\text{-}Auth}(ATM)) &= 4 \\ n(PLATP_{MP\text{-}Auth}) &= 2 \end{aligned}$$

Hence, the full threat model chart has $4^2 \cdot 2^1 = 32$ lines. The threat models considered in previous work [133] can be reinterpreted in my chart as shown in Table 4.6.

	Human	*ATM*	*ATP*
(a)	*normal*	*normal*	*malicious*

Table 4.6 A scenario for the Cash-point ceremony.

However, my chart also highlights at least three more relevant threat models that haven't been considered yet. These are summarised in Table 4.7.

	Human	*ATM*	*ATP*
(b)	*error*	*normal*	*malicious*
(c)	*choice*	*normal*	*normal*
(d)	*error+choice*	*bug*	*normal*

Table 4.7 Other interesting scenarios for the Cash-point ceremony.

The threat model (a) analysed in [133] considers a human that behaves normally while he is interacting with an ATM, also acting following the protocol. However, the system is under attack of an *ATP*. I believe that it would be interesting to consider also threat model (b), where the human simply makes mistakes, potentially facilitating the *ATP*'s attack; threat model (c) in order to check what could be the repercussions for the system when a human interacts freely, choosing some actions; and threat model (d), where the human not only makes mistakes but decides to perform some actions, in presence of a bugged ATM that potentially tries to get some personal profit, becoming itself an attacker against the ATM system.

4.6 The Danish Mobilpendlerkort

As a more detailed example, let us consider the inspection ceremony for the mobile transport ticket in Denmark, which is known to have two vulnerabilities [68]. The Danish ticket inspection ceremony considers five principals, two of which are human beings (ticket holder and ticket inspector). Despite the combinatorial explosion due to the number of principals and actions, it is interesting to dissect some combinations, especially those in which both human principals deviate from the prescribed ceremony.

4.6.1 Description

As a precondition, the human downloads on his phone a specific app, called *Mobilpendlerkort*, which allows the human to buy a ticket using a credit card. The human gives the phone his personal details and travelling preferences, which the phone then forwards to the train company's server. This server sends back to the phone a QR code that encodes a signed version of the human's travelling preferences. Upon request of the ticket inspector, the human shows the phone with the QR code, and the inspector

visually checks the authenticity of the ticket. Then, the inspector checks the validity of the barcode via a ticket scanner, which has access to the verification key needed to validate the signature on the QR code. Finally, the scanner emits a green light if the verification succeeds, a red light otherwise. Additionally, the inspector may ask the human to show a valid ID document to check the human's identity.

I identify the following principals in the ceremony (where, for simplicity, I have considered the phone and the ticketing app as a single technical system):

$$\begin{aligned}
&\mathit{Humans}(\mathit{Danish\text{-}Mobil}) := \{\mathit{Human}, \mathit{Inspector}\}\\
&\mathit{Technicals}(\mathit{Danish\text{-}Mobil}) :=\\
&\qquad\qquad \{\mathit{Phone}, \mathit{Scanner}, \mathit{Server}\}\\
&\mathit{Principals}(\mathit{Danish\text{-}Mobil}) :=\\
&\qquad\qquad \mathit{Humans}(\mathit{Danish\text{-}Mobil}) \cup\\
&\qquad\qquad \mathit{Technicals}(\mathit{Danish\text{-}Mobil}) \cup\\
&\qquad\qquad \mathit{ATP}(\mathit{Danish\text{-}Mobil})
\end{aligned}$$

The most general case with legitimate principals getting all possible labels sees:

$$\begin{aligned}
n(\mathit{PLHumans}_{\mathit{Danish\text{-}Mobil}}(\mathit{Human})) &= 4\\
n(\mathit{PLHumans}_{\mathit{Danish\text{-}Mobil}}(\mathit{Inspector})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Danish\text{-}Mobil}}(\mathit{Phone})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Danish\text{-}Mobil}}(\mathit{Scanner})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Danish\text{-}Mobil}}(\mathit{Server})) &= 4
\end{aligned}$$

I also assume the attacking third party to only misbehave as formalised by its two labels, therefore:

$$n(\mathit{PLATP}_{\mathit{Danish\text{-}Mobil}}) = 2$$

Hence, the full threat model chart has $4^5 \cdot 2^1 = 2048$ lines.

	Human	*Phone*	*Scanner*	*Server*	*Inspector*	*ATP*
(a)	*choice*	*malicious*	*normal*	*normal*	*choice*	—
(b)	*choice*	*malicious*	*normal*	*normal*	*normal*	*choice*

Table 4.8 The two threat models already considered for the Danish-Mobil ceremony.

	Human	*Phone*	*Scanner*	*Server*	*Inspector*	*ATP*
(c)	*normal*	*malicious*	*normal*	*normal*	*normal*	*malicious*

Table 4.9 An additional threat model relevant for the Danish-Mobil ceremony.

4.6.2 Threat Models

I focus on three threat models derived from my chart. The threat models (a) and (b) in Table 4.8 have been considered in [68] and shown to lead to vulnerabilities. The additional threat model (c) in Table 4.9 provides new insights about a potential yet realistic attack to the ceremony.

More in detail, threat model (a) considers a human who chooses to forge the screen of the app that displays the ticket details. The human chooses *not* to use the original app, but installs a fake app on his phone so as to mimic both watermark and background of the original app. This threat model sees the inspector only choose to visually check the ticket details and *not* to scan the signed QR code. This is routine in Metro or local trains [68]. This vulnerability notably doesn't require an attacking third party because the human takes advance of an inspector who chooses to deviate from the ceremony. The attack is possible because the ceremony is designed to let the human (i.e., the ticket inspector I) decide whether a ticket is genuine or not.

Threat model (b) differs from the previous one as the inspector doesn't deviate from the protocol and there is an attacking third party. The latter plays the role of a ticket holder who buys a valid ticket but then chooses to send the signed QR code to the human. Then, the human uses a fake app as in (a) to mimic watermark

and background of the original app and display the QR code sent by the attacking third party. Notably, both the scanner and the inspector follow the ceremony. This vulnerability, hence, enables a ticket holder to buy a valid ticket that can be shared with other people to travel for free. More specifically, the attack is possible because the QR code doesn't include the personal details of the ticket holder. Thus, a valid QR code can be reused by different people. The attacking third party is essential to signify the attack and provides a way to reason about collusion among different principals in a quite abstract and general manner.

Threat model (c) has all principals except the phone behave as *normal* and also features an attacking third party active towards the phone, hence behaving as *malicious*. This threat model represents a form of collusion of the attacking third party with the human's phone, for example with the phone running some malware on behalf of the attacking party. As a consequence, the phone is enabled to steal a valid QR code from a ticket bought by the human and send it back to the attacking party, who, in turn, can redistribute tickets to colluding phones.

Threat model (c) has not been considered so far and arises here thanks to my charting method. To demonstrate the relevance of the chart I analysed threat model (c) using the formal and automated tool Tamarin although it is important to highlight that my method can be used with any tool for the analysis of security protocols and ceremonies.

4.6.3 Formal Analysis

As explained in Section 2.3, a Tamarin rule has a premise and a conclusion and operates on a multiset of facts. Facts are predicates that store state information. Executing a rule means that all facts in the premise are present in the current state and are consumed in favour of the facts in the conclusion. Tamarin supports *linear facts*, which

may be consumed by rules only once, and *persistent facts*, which may be consumed by rules arbitrarily often.

A fundamental choice is to exclude Tamarin's built-in attacker model (i.e., the Dolev-Yao attacker) in favour of the threats provided by the line in my chart model. More specifically, the communication among principals is modelled by non-persistent private channel rules. Human, Scanner, Server, and Inspector are labelled as *normal*, hence their Tamarin specifications are the ones prescribed by the ceremony. Phone and ATP are labelled as *malicious*, hence they have deviating behaviours which are captured by the Tamarin rules as shown in Listing 4.1. The full formal model is available in Section A.2 and online here [146].

```
rule Pleaks:
     [!Pstore($P,'P_5',<s1, signed_qr>)]
     --[PleakstoATP('ticket',signed_qr)]->
     [Out_S($P, $ATP,'leak_ticket',signed_qr)]


rule Pfw:
     [In_S($ATP,$P,'fake_ticket',signed_qr_atp),
      !Pstore($P,'P_5',<s1, signed_qr>)]
     --[Pgetsfake('ticket',s1,signed_qr_atp)]->
     [Out_S($P, $H,'fw_ticket',<s1,signed_qr_atp>)]


rule ATPgets:
     [In_S($P,$ATP,'leak_ticket',signed_qr)]
     --[ATPKnow($P,signed_qr)]->
     [!ATPK($P,signed_qr)]
```

```
rule ATPsends:
     [!ATPK($P,signed_qr)]
     -->
     [Out_S($ATP, $PP,'fake_ticket',signed_qr)]
```

Listing 4.1 Rules of the Danish Mobilpendlerkort ceremony

The fact `!Pstore` formalises a phone storing the ticket purchased by the traveller, while the fact `!ATPK` expresses the knowledge of the ATP, and is conveniently instantiated to a signed QR code. Rules `Pleaks` and `Pfw` model behaviours of a malicious phone that respectively leaks a signed QR code (`signed_qr`) to the ATP and forwards another signed QR code sent by the ATP (`signed_qr_atp`) to the human. Rules `ATPgets` and `ATPsends` model behaviours of a malicious ATP that respectively receives a signed QR code from a malicious phone and forwards it to a *different* malicious phone. I argue that these rules reflect very basic malicious behaviours for phone and ATP. I check one of the main security properties for a ticketing ceremony, namely, that two different human beings cannot ride with the same ticket. This is formalised in Tamarin by the lemma in Listing 4.2:

```
lemma oneticketpertraveller:
      "All h1 h2 s #i #j. OK(h1,s)@i &
       OK(h2,s)@j & i<j ==> h1=h2"
```

Listing 4.2 Lemma 'oneticketpertraveller' for the Danish Mobilpendlerkort ceremony

The label `OK(h,s)` expresses a successful verification of a ticket with serial number `s` and ticket holder `h` by a ticket inspector. Thus, the property says that two distinct

verifications of a ticket with the same serial number should concern the same ticket holder.

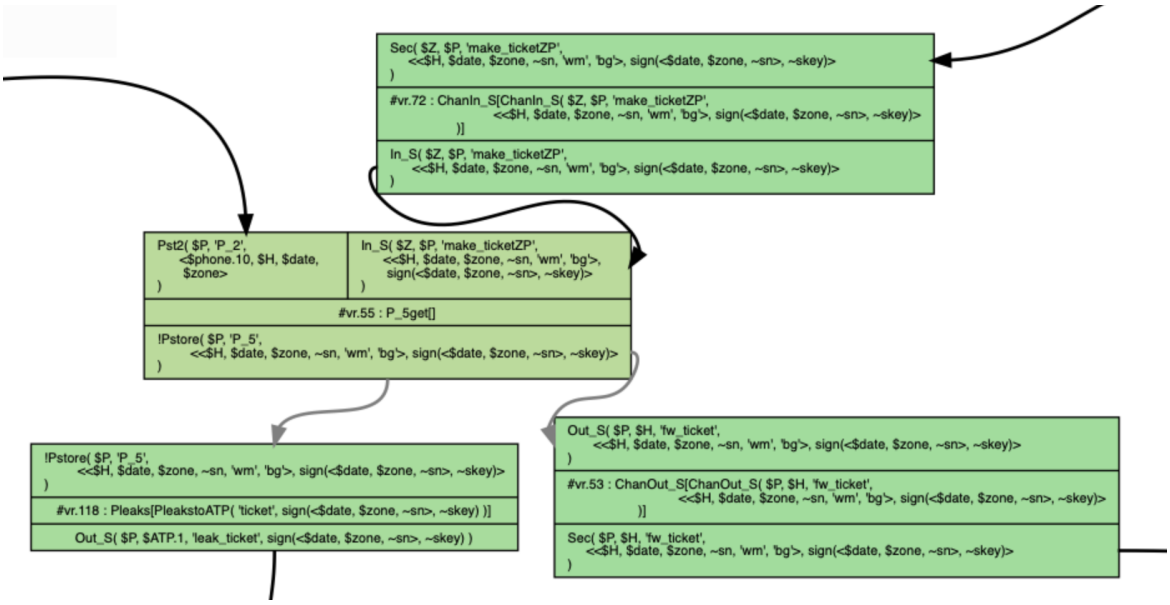


Figure 4.1 A snapshot of a portion of the attack graph provided by Tamarin. The phone leaks part of the ticket to the ATP

Tamarin finds an attack that violates the stated property and provides a graph that details the steps of the attack to break the property. I focus on two crucial steps.

The first step is outlined in Figure 4.1, in which the conclusion of rule `P_5get` is consumed by the premises of rules `Pleaks` and `ChanOut_S`, thus creating two branches. The former rule leaks the signed QR code to the ATP, while the latter forwards the ticket to the human, who notably doesn't detect that something bad is happening.

The other crucial step is outlined in Figure 4.2, which depicts a portion of what follows in the branch due to the rule `Pleaks`. One of the premises of the rule `Pfw` is satisfied by a signed QR code that was purchased by another traveller but is rerouted by the ATP through the malicious phones. The other premise is satisfied by the fact `!Pstore`, which stores the correct ticket details of the current traveller. The malicious phone combines the correct ticket details with the signed QR code forwarded

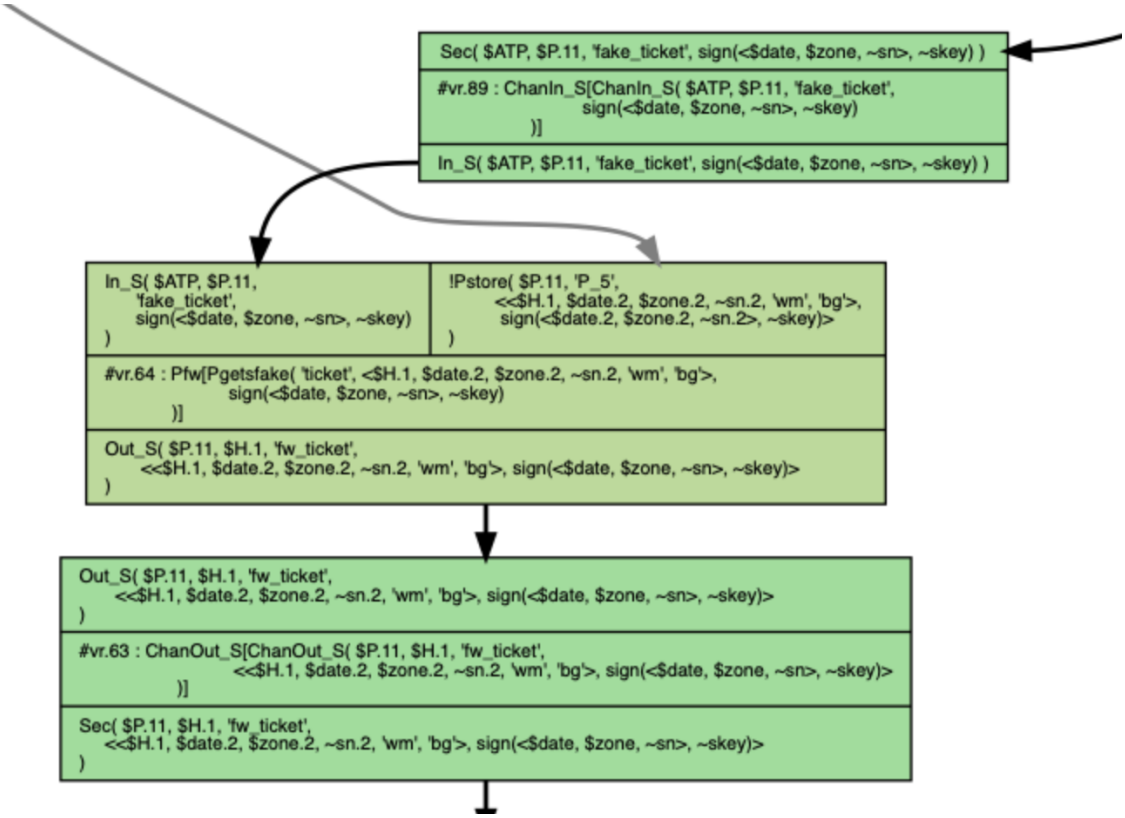


Figure 4.2 A snapshot of a portion of the attack graph provided by Tamarin. The phone combines the details of the correct ticket with the signed QR code provided by the ATP

by the ATP, and forwards the resulting fake ticket to the human. Later, during ticket inspection, the fake ticket is successfully verified.

Although the attack mechanism is similar to the one described in the previous threat model, the attacking party doesn't need to create a fake ticket but just to redistribute the signed QR code from one malicious phone to another. The consequences on the ticket holder are more serious in this case. The holder is unaware that his ticket is being reused by someone else. However, post-analysis techniques may reveal a misuse of fake tickets, hence suggest a software update to the scanner so that the scanner

would invalidate the fake ticket (further legal action against the ticket holder could follow).

It is clear that the full threat model chart allows us to appreciate how the same overarching attack mechanism may be used in different scenarios according to the selected threat model. As we have seen, different consequences may arise, thus bringing novel insights to a focus.

4.7 Conclusions

The method that I proposed allows one to build the chart of all threat models for a given security ceremony upon the basis of the relevant principals (thus identifying the columns of the table) and their capabilities (thus identifying the rows of the table). As I remarked above, the chart helps a security analyst capture scenarios that would not be captured by simply applying the standard Dolev-Yao attacker. The security analyst can then relate the threat models that have already been considered, if any, to those that haven't and, in every case, address all possible threat models of interest pragmatically.

The methodology has been successfully applied on four ceremonies, suggesting some interesting threats that haven't been investigated although they are well worth of scrutiny. Also, I have demonstrated the relevance of the chart, modelling and analysing one of the threat models discovered through this methodology, showing that I was able to uncover, using model checking, a new security threat that affected one of the security ceremonies considered.

Aspects of "beautification" from Chapter 3 are not considered here since they are meant to be for security ceremonies and not for threat models.

I believe that my formalisation of the ceremony principals based on their actions provides a semantics that is, in some sense, more operational than the one that is

traditionally considered for security protocols. For example, notions such as impersonation and collusion are neatly represented through my approach by endowing principals with appropriate functioning or behaviour. Still, I am aware that the notions that I have introduced in this work are quite coarse and in need of a more fine-grained formalisation.

The research ahead of us is clearly defined: how to cope with the size and complexities of the full threat model chart. One approach could be the description of appropriate measures and weights to prioritise the different threat models so as to create an ordered list. The given ceremony could then be verified, in turn, against each item of the list.

Another approach could be the definition of methods to handle the full threat model chart in one go, perhaps parameterising the findings upon the threat model. Tamarin can be customised to dealing with a specific threat model extracted from the chart, hence disposing with the traditional Dolev-Yao attacker model; however, it is not yet clear whether and how Tamarin could scale up to the challenges identified here. These challenges stretch out the requirements traditionally put on tool support, because the formal analysis will have to identify not only an attack but also the threat models that substantiate it.

Chapter 5

X-Men: A Mutation-Based Approach for the Formal Analysis of Security Ceremonies

This chapter introduces a novel approach for the formal analysis of security ceremonies that focuses on the vulnerabilities that result from the mistakes that human users might make. This work has been published as: Sempreboni, D. and Viganò, L. (2020). X-Men: A Mutation-Based Approach for the Formal Analysis of Security Ceremonies, In *Proceedings of the 5th IEEE European Symposium on Security and Privacy, EuroS&P*, IEEE. [151].

5.1 Motivation

As widely discussed in Chapter 2, *security ceremony analysis*, in contrast to security protocol analysis, for which a plethora of mature approaches and tools exist, is a

discipline that is still in its childhood, with no widely recognized methodologies or comprehensive toolsets.

State-of-the-art approaches and tools for security protocol analysis (e.g., [5, 28, 61, 108, 124]) cannot be directly employed for security ceremonies as they take a "black&white" view and formalise protocols by

- considering one or more attackers that can carry out whatever actions they are able to in order to attack the protocol, but then
- modelling all other protocols actors (regardless of whether they are computers or human users) as honest processes that behave according to the protocol specification.

When considering security ceremonies, in which humans are first-class actors, it is not enough to take this "black&white" view. It is not enough to model human users as "honest processes" or as attackers, because they are neither. Modelling a person's behaviour is not simple and requires formalising the human "shades of grey" that such approaches are not able to express nor reason about. It requires modelling the way humans interact with the protocols, their behaviour and the mistakes they may make, independent of attacks and, in fact, independent of the presence of an attacker.

Some preliminary approaches have been proposed for security ceremony analysis (e.g., [15, 16, 20, 36, 50, 84, 104, 131, 130]), but they have barely skimmed the surface of taking into account human behavioural and cognitive aspects in their relation with "machine" security.

5.2 Contributions

In this chapter, I provide three main contributions.

1. Formalization I define a formal approach that allows security analysts to model possible mistakes by human users as *mutations* with respect to the behaviour that the ceremony originally specified for such users. I focus on three main human mutations of a ceremony,

- *skipping one or more of the actions that the ceremony expects the human user to carry out* (such as sending or receiving a message),
- *replacing a message with another one*,
- *adding an action*,

and their combinations (but my approach is open to extensions with other mutations as also shown in Chapter 6).

Human ceremony mutations will likely have an effect also on the other agents of the ceremony, honest or malicious as they may be. There are two cases: (1) the other agents are able to reply to a human mutation because the changes are not too relevant or because the ceremony has somehow made provision for it (e.g., by an if-the-else that captures both original and mutated human behaviour), or (2) the other agents are not able to reply to a human mutation. To investigate whether this human mutation may lead to an attack, I formalise *algorithms for human mutations* and *algorithms for matching mutations* for the other agents, which allow us to create a complete mutated ceremony specification that can be executed and analysed for vulnerabilities. My algorithms allow for the analysis of the original ceremony specification and its possible mutations, which may include the way in which the ceremony has actually been implemented.[1]

[1]It is often the case that the implementation of a protocol or ceremony deviates from the original specification. There are several possible reasons for this. For instance, the implementation might have deviated from the specification in order to accommodate initially unforeseen behaviour by the human users (and this might actually be one of the reasons for the issues in my first case study, the Oyster ceremony) or simply because the implementers did some mistakes (as in my second case study, the SAML-based Single Sign-on for Google Apps [7]).

2. Tool I have developed a prototype tool called *X-Men* (the name was chosen to suggest that it considers human mutations), which, as shown in Figure 5.1, creates mutated models that can then be input to Tamarin [108], one of the most advanced tools for the automatic unbounded verification of security protocols. X-Men can be used with human mutations only (without matching) but this will often yield non-executable specifications as the non-human roles simply won't reply (thus thwarting attacks caused by the human mutation). Matching mutations adjust non-human roles so that they can be executed together with a mutated human role. They are implemented automatically by X-Men, which generates the matching mutation from the protocol specification based on the human mutation (it changes the non-human-role specification to receive/send messages according to the human mutation) and propagates mutations to create an executable trace that can be analysed in search for attacks. These attacks might be real attacks on the ceremony's (specification and) implementation, or be just the result of the mutations and not be applicable on the actual implementation. In the spirit of *mutation testing* [55, 83, 33, 51], the attacks discovered by X-Men could be used to generate and apply test cases for the ceremony implementation, but I leave this extension of X-Men for future work.

3. Proof-of-concept I have applied my approach to two real-life case studies, the Oyster ceremony and the SAML-based Single Sign-on for Google Apps [7], uncovering a number of concrete vulnerabilities, which had so far been discovered only by empirical observation of the actual ceremony execution or by directly formalising alternative specifications of the ceremony by hand. Instead, X-Men allowed me to generate them automatically.

5.3 Outline

In Section 5.4, I introduce my motivating and running example. In Section 5.5, I describe the intuitions that underlie my approach. In Section 5.6, I describe how I formally model security ceremonies and their mutations. In Section 5.8, I describe X-Men and its proof-of-concept. In Section 5.9, I discuss related work. In Section 5.10, I draw conclusions and discuss future work. All my formal models and the code of X-Men are available online at [178]. The formal models are also available in Section A.3 and in Section A.4.

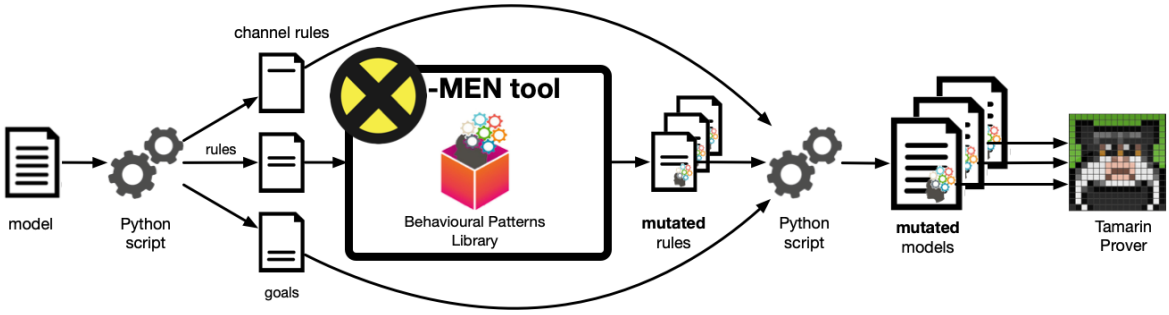


Figure 5.1 The workflow of the X-Men tool: from models to mutated models that are input to Tamarin

5.4 An Example: The Oyster Card Ceremony

I will use the Oyster Card ceremony as a motivating and running example. The *Oyster Card* (or just Oyster, for short) is a plastic credit-card-sized, rechargeable, stored-value, contactless smart card used on public transport in Greater London in the United Kingdom. The Oyster is a form of electronic ticket that can hold pay-as-you-go credit, travel cards and passes for underground and overground trains, buses and trams. It is promoted by *Transport for London (TfL)* and since its introduction in June 2003, more than 86 million cards have been used [165]. Similar systems are in use in a large number of other countries in almost all continents, such as France, Italy, Denmark,

Finland, South Africa, Argentina, Chile, Australia and Japan, and, interestingly, most of them suffer from problems similar to the ones of the Oyster that I will discuss in the next sections.

As shown in Figure 5.2a, the Oyster is used by touching it on an electronic reader when entering and leaving the transport system in order to validate it or deduct funds. Actually, this touch-in/touch-out is part of the ceremony used on the London underground (nicknamed the *Tube*) and trains, which is what I focus on in this chapter, whereas on London buses passengers touch in their Oyster only when boarding (instead, in Sydney, Australia, passengers are required also to touch out when they alight the bus). Figure 5.2b shows an entrance/exit gate of the Tube.

(a) Touching the Oyster on an Electronic Card Reader

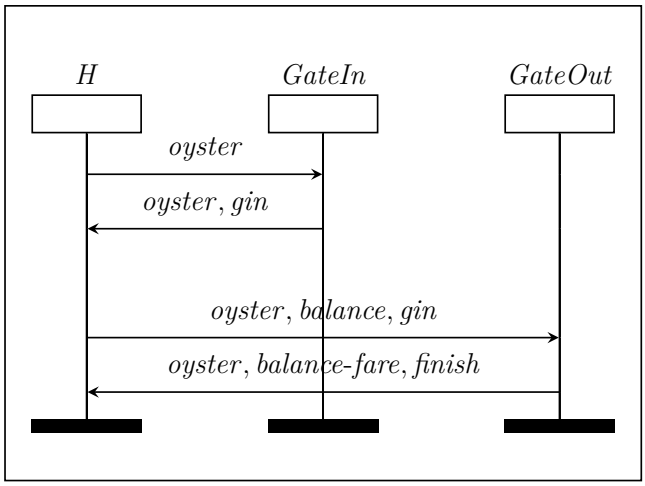
(b) A Gate of the Tube

Figure 5.2 Using the Oyster Card in the Tube

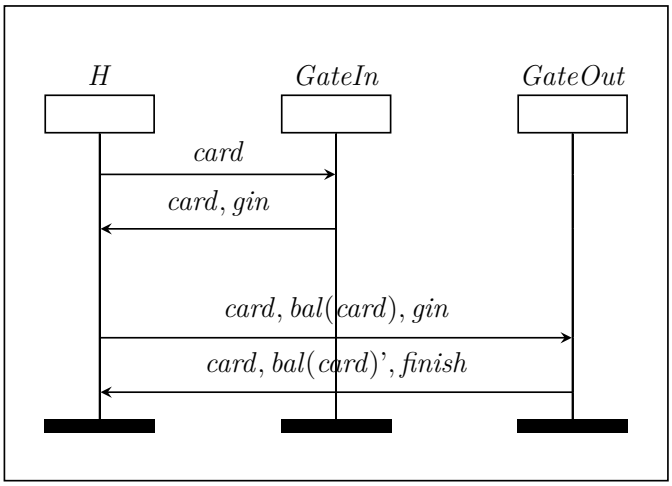
Figure 5.3a gives a Message Sequence Chart (MSC) of the main Oyster Ceremony for the Tube, which is carried out by 3 roles: the human passenger H, the entrance gate *GateIn* and the exit gate *GateOut*.

1. The human passenger H touches their Oyster on the reader at the entrance gate, which amounts to H sending the Oyster number *oyster* to *GateIn*.

2. The reader writes an identifier on the Oyster, which amounts to *GateIn* replying with the message *oyster*, *gin*, where *gin* is the identifier of *GateIn*.



(a) The Main Oyster Ceremony for the Tube



(b) The Generalised Main Ceremony for the Tube

Figure 5.3 The Ceremonies for the Tube

3. At the end of the journey, the passenger touches the Oyster on the reader at the exit gate, which amounts to H sending to *GateOut* the number *oyster*, the current *balance* of the card and *gin*.

4. *GateOut* calculates the journey fare based on the distance travelled from *GateIn*, subtracts the amount from the card's balance, and sends to H the new balance along with the card number and a *finish* flag.

Some remarks are in order. First of all, note that I did not obtain this specification from TfL, with whom I have not been in touch, but rather I modelled my own experience of using the Oyster. This is fine as I do not need my example to be real but rather realistic enough to showcase the main features of my approach; still, the vulnerabilities that I identify are actual problems that the real Oyster system suffers from.

Second, even though the Oyster is based on the MiFare chip, which in its first version (Mifare Classic family) used the proprietary encryption algorithm Crypto-1, my specification does not use any kind of encryption for the messages. This does not represent a lack of accuracy as I actually aim to model the ceremony in a way that is independent of the low-level cryptographic details, thereby also keeping in mind that my approach focuses on what is under direct influence and control of the human, and cryptography most likely is not. However, it would not be difficult to include encryption and decryption in my specifications, and in fact the language that I describe below does contain cryptographic operators.[2]

Third, I focused only on the core message-passing of the ceremony and did not include the information that is displayed on the screens that are placed above the gate's reader, which show, e.g., the credit on the card when entering and exiting and the fare of the trip when exiting.

Fourth, the ceremony in Figure 5.3a is actually one of the possible ceremonies that could be considered for the use of the Oyster and several variants could be modelled, such as: a ceremony in which the reader at the exit gate does not immediately synchronize with the system, a ceremony in which the passenger does not have enough credit for the entrance gate to open (if the Oyster's balance is too low, the gate would display a message to the passenger asking them to top up the credit on the card), or a ceremony in which the passenger changes from an overground train to an underground

[2]Note also that initial versions of the MiFare chip, and thus of the Oyster, suffered from a number of attacks [66, 53, 46], but the current version of the Oyster does not suffer from these problems any more since it is based on the new MiFare DESFire family that uses stronger encryption algorithms.

train or vice versa, and thereby touches the Oyster at an intermediate gate to register the change of train. Again, I aim to be realistic rather than real and, in fact, my approach generalises to these variants quite straightforwardly.

Finally, passengers are nowadays able to pay not only with the Oyster but also with a contactless credit or debit card (possibly associated with an Apple Pay or Google Pay device). In that case, the ceremony is the same as the one in Figure 5.3a but without the *balance* and replacing *oyster* with the number of the contactless credit/debit card (the physical one used to touch in/out or the one associated with Apple or Google Pay). To avoid having to distinguish the two cases, let me introduce a generalised ceremony for the Tube, which passengers can carry out with either their Oyster or a contactless card, as shown in Figure 5.3b. Here, I use a public unary function *bal* that computes the current balance of an Oyster or simply sends a message "*accept*" in case of a contactless card. This is what H sends in the third message, and then *GateOut* replies in the final message by sending $bal(card)'$, which is the updated balance of the Oyster or another "*accept*" message, respectively.

Before I continue with the discussion of how I formally model security ceremonies in my approach, let us return to Figure 5.2a, where the sticker beside the reader reminds passengers to always touch in and out. In fact, the London underground is quite full of posters like the ones in Figure 5.4. The poster on the left of Figure 5.4 reminds passengers that in order to pay the right fare, they need to touch in at the start and touch out at the end of all journeys; if they do not, then TfL will not know where the passenger has travelled, so they cannot charge the right fare for the journey. This is called an *incomplete journey* and the passenger could be charged a maximum fare ranging between £8.00 and £19.80 [164]. Passengers who do not touch in at the start of a journey are also liable to pay a penalty fare (or could even be prosecuted).

Figure 5.4 Warnings issued to the Tube passengers

The poster on the right of Figure 5.4 warns passengers that if they touch on a reader their purse or wallet containing two or more cards (be they Oyster cards or contactless payment cards), then they could experience *card clash* [163]. This means that when the card reader detects two cards, it could take payment from a card that the passenger did not intend to pay with, or, more dangerously, that the passenger could be charged two fares for his journey or even two maximum fares for his journey (this happens when a passenger mistakenly touches in with one card and touches out with another card, resulting in two incomplete journeys).

It is interesting to observe that, in both these cases, security is "pushed" from the system to the human user. But humans do mistakes and this might endanger their security, which here means that they possibly have to pay considerably more than they should. My approach allows us to show (in a formal and automated way) that indeed if passengers forget to touch in or out, or touch with two or more cards at the

same time, then they will be billed unfairly. Let me thus proceed by explaining how I formally model and reason about security ceremonies.

5.5 My Approach in a Nutshell

The standard way to formally model and analyse a security protocol/ceremony is to formalise how agents (attempt to) execute the *roles* of the protocol/ceremony to achieve one or more security goals in the presence of an attacker. Roles are sequences of events (sending or receiving messages, generating fresh values, etc.), which are usually represented graphically by a structure generated by causal interaction such as *strands* [63] or the vertical lines in MSCs and *Alice&Bob notation* [4], or less graphically by a process in a process algebra such as in the *applied pi calculus* [1]. In Tamarin (and thus in the X-Men tool), a role is formalised by a so-called *role script*, which is basically the projection to an individual role of an extended Alice&Bob specification, and corresponds to a strand or an applied pi calculus process.

I can represent this graphically by viewing the roles/strands of a ceremony as separate lines of assembled jigsaw puzzle pieces that can be connected with each other as shown in the example in Figure 5.5. When complete, the jigsaw puzzle produces a complete picture: the run of the ceremony.

Now, we have all been there: you are trying to assemble one of those really difficult jigsaw puzzles, you know, one of those where the resulting image is so complex that it is difficult to understand which pieces you should actually interlock. You start from the borders, trying to complete at least one line and proceed from there, but even that is proving to be difficult as you do not understand which pieces do really fit together. So, what do you do? You try. You try to interlock pieces that appear to fit together even though this will turn out to be wrong as they will not allow you to produce the

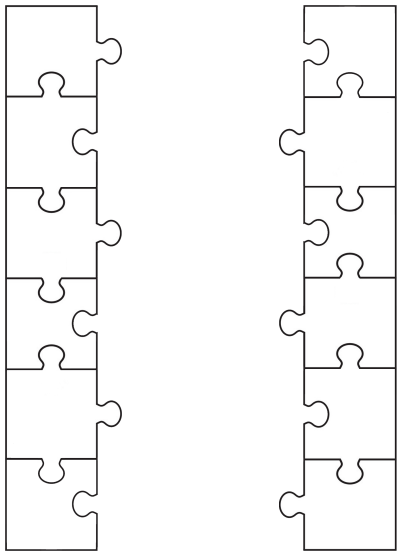


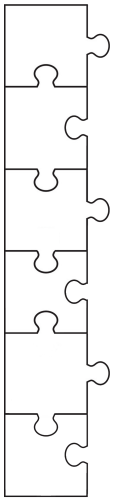
Figure 5.5 A simple ceremony between a User (left) and a System (right) depicted as a jigsaw puzzle

desired image — but you do not know that yet. Or maybe you simply do a mistake and append a piece that does not belong there.

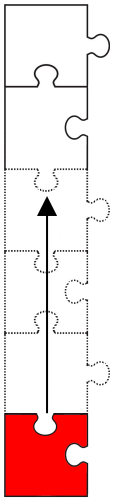
This is illustrated by Figure 5.6: the human user could append a wrong piece pictured in red as in Figure 5.6c, which raises the question of how the two remaining pieces would fit (they are thus drawn with dotted lines), or the human could not know how long the edge should be and terminate it by attaching the piece pictured in red as in Figure 5.6b; or the human could add one more piece to the edge as in Figure 5.6d.

Returning to my running example, the human user might not fully understand the ceremony role that he is supposed to carry out and

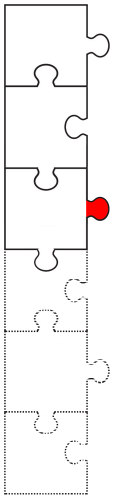
- *skip* some intermediate actions, e.g., touching out with an Oyster without having touched in with any card, as illustrated in Figure 5.6b by the anticipated termination of the role;
- *replace* an action with another, e.g., using a contactless credit card to touch out instead of the Oyster he used to touch in, as illustrated in Figure 5.6c by the different outgoing connector, which represents a different message being sent;



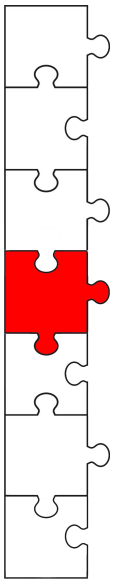
(a) The role User as it was specified



(b) The role User as carried out by a human who connects a piece to terminate the role sooner than specified



(c) The role User as carried out by a human who connects a different piece than the one specified



(d) The role User as carried out by a human who connects a piece to extend the length of the role

Figure 5.6 A human carrying out the role User... and mutating it, by mistake or lack of understanding

- *add* some actions, e.g., touch in with two cards, as illustrated in Figure 5.6d by the additional piece.

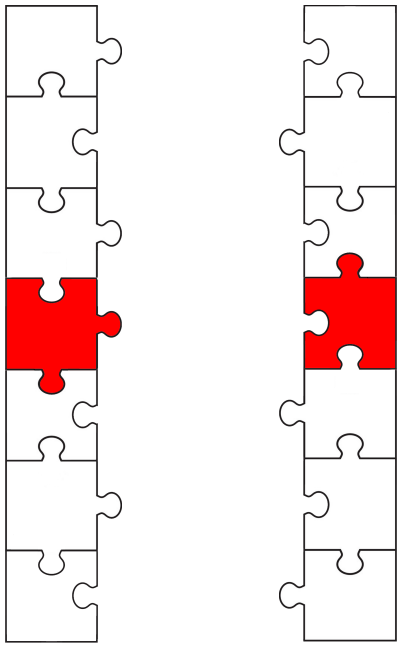


Figure 5.7 The "add" mutation of the role User (as in Figure 5.6d) and the matching mutation of the role System

In my approach, I represent these human "mistakes" as *mutations* with respect to the role as specified originally — hence the name "X-Men" for my tool, which captures the fact that I am considering mutations of the original human behaviour. Such a mutation does not just have a local effect (for that event of the role) but will likely have an effect on the subsequent events in the role, which I illustrated by drawing the subsequent puzzle pieces with dotted lines. This is because the knowledge of the human agent will likely change depending on what has really happened.

It is, however, not enough to simply allow the human to carry out these unforeseen actions (add, skip or replace some parts of the role). In order to reason about what would happen if the human carried out these mutations, I need to capture the fact that a mutation of the human behaviour will likely have an effect also on the other agents of the ceremony. More specifically, consider again, for simplicity, the ceremony between User and System in Figure 5.5 and consider the scenario in which a human playing the role User replaces an event of his role with a different one, i.e., sends a

message m' instead of the specified message m, as depicted in Figure 5.6c, which is a mutation of Figure 5.6a. There are two cases.

In the first case, the System is able to reply to m'. This means that the System can still receive (and "understand") and reply to m' because the changes with respect to m are not too relevant. For instance, this might happen when the ceremony does not provide the System with enough information to check the content of m', e.g., when the User sends a contactless card number instead of an Oyster card number but the System does not have previous information that allows it to check whether it received the correct card number, or when the message has been encrypted with a symmetric key that the System does not (yet) possess. In this case, we can carry on with our analysis of the ceremony to check whether either the original or the mutated User role lead to an attack.

In the second case, the System is not able to reply to m' as that mutation is not envisioned by the System's role as specified by the original ceremony. But what about the ceremony's implementation? Does the implementation really conform to the specification? If it does, then the implementation of the System role will not reply and we are fine as the run with the mutation m' will not terminate. But what if the ceremony's developers, after they designed the specification and/or deployed the implementation, realised that the User could indeed send a different message (or skip some actions or add some) and made provisions for this case? For instance, they could have introduced in the implementation an "if-then-else" that captures both m and m', i.e.: "if you receive m then reply with message n else if you receive an $m' \neq m$ then reply with message n'".[3] To reason about such a situation, we can use the mutation as

[3]Note that this does not mean that the User is fully aware of this. The User might just be aware of (or have been instructed about) the "then branch" of the System's role, which captures the User's normal behaviour; think of the Oyster User who follows the touch-in-touch-out ceremony as expected. Hence, the User might, unknowingly and unwillingly, fall into the "else branch" of the System's role (e.g., by touching out with a contactless card instead of the Oyster card that was used for touch in) and thus be billed much more than expected. The problem with these "else branches" is that they often were not present in the original specification of the whole ceremony and were added to the

a test case that is relevant for the ceremony's implementation. We pair the mutation of the User role with a *matching mutation* of the System role to generate an executable trace of the ceremony. This is in line with *mutation testing* [55, 83, 33, 51], which is an approach to design software tests where mutants are based on well-defined mutation operators that either mimic typical programming errors (such as using the wrong operator or variable name) or force the creation of valuable tests (such as dividing each expression by zero). In my approach, mutants are based on mutation operators that mimic typical human mistakes (add, skip or replace, as discussed in this section and in Section 5.2) and force the creation of mutations in the other ceremony agents to match the human mutation. For concreteness, for the ceremony between User and System in Figure 5.5, my approach mutates the role of the System as shown in Figure 5.7 to match the human User's replace mutation of Figure 5.6c. The mutation of the step of the System to match the mutated step of the human User possibly entails a mutation of the subsequent steps of the System role, which I again illustrate with dotted lines.

If these matching mutations lead to an attack, then I can check with the ceremony designers whether the mutated specification makes sense and, in any case, use the obtained attack trace to generate concrete tests cases to be applied to the ceremony's implementation. This will allow us to check whether the attack entailed by the mutations is a false positive or a real attack.

The scenarios for the other human mutations and their matching mutations are similar. So, summarising, my approach takes as input the specification of a ceremony and the goal(s) it should achieve, and then generates both mutations of the human

implementation as an afterthought, after having observed the "wrong" behaviour of users, as was likely the case for the Oyster ceremony. Warnings like the ones in Figure 5.4 are meant to alert the users about the "else branch" of the ceremony. I believe that rather than adding the "else branch", it would have been better to change (the specification and) the implementation of the ceremony to forbid these mistakes (e.g., by programming the gates to warn the users that they are touching with the wrong card or with two cards), but I recognize that this might not always be possible, especially if all the software and hardware components of the System have already been deployed and installed.

agent's role (allowing him to add, skip or replace actions) and the matching mutations of the other roles of the ceremony. The resulting mutated ceremony specifications are then fed into Tamarin to search for attacks. I leave the step of concretising the attack traces found into test cases as future work (although I expect this to be not too difficult by proceeding along the lines of [172, 127]).

5.6 Formal Modelling of Security Ceremonies

I adopt, adapt and extend notions that are used in most of the state-of-the-art approaches and tools for the formal analysis of security protocols. For concreteness, my tool X-Men extends the Tamarin prover [16, 15, 108] to model and analyse security ceremonies with mutations caused by human users, but my approach is general and independent of Tamarin and could be applied similarly to other tools such as ProVerif [28], Maude-NPA [61], AVANTSSAR [5] and its follow-up SPaCIoS [172].

I first summarize some basic notions (importing them from papers in which Tamarin is presented and used) and then discuss the formal specification of ceremonies, the execution model, the modelling of human agents, and the security goals.

5.6.1 Messages

The *term algebra of messages* is given by $\mathcal{T}_\Sigma(\mathcal{V})$, where Σ is a signature and $\mathcal{V}$ is a disjoint, countably infinite set of variables. A term m is *ground* when it contains no variables. $F_{sym} \subset \Sigma$ denotes a finite set of function symbols that contains function symbols for: the *pairing* $pair(m_1, m_2)$ of two messages m_1 and m_2, also denoted by $\langle m_1, m_2 \rangle$, where, for brevity, I write, e.g., $\langle m_1, m_2, m_3 \rangle$ for $\langle m_1, \langle m_2, m_3 \rangle \rangle$; the *first projection* $\pi_1(m)$ and *second projection* $\pi_2(m)$ of a pair m of terms; the *hash* $h(m)$ of a term m; the *symmetric encryption* $senc(m, k)$ and the *symmetric decryption* $sdec(m, k)$

of m with k; the *asymmetric encryption* $aenc(m, k)$ and the *asymmetric decryption* $adec(m, k)$ of m with k; the *signature* $sign(m, k)$ and the corresponding *verification* $verify(sign(m, k_1), m, k_2)$. The function $pk(k)$ represents the public key corresponding to the private key k.

Messages are composed and decomposed using the standard Dolev-Yao-style equational theory for these functions, based on the equations

- $\pi_1(\langle m_1, m_2\rangle) = m_1$ and $\pi_2(\langle m_1, m_2\rangle) = m_2$,
- $sdec(senc(m_1, m_2), m_2) = m_1$,
- $adec(aenc(m_1, pk(m_2), m_2) = m_1$,
- $verify(sign(m, k), m, pk(k)) = true$.

However, as I do for the Oyster ceremony, my approach allows me also not to consider explicitly the presence of a (Dolev-Yao) attacker and focus on capturing the way human agents might interact insecurely with the other ceremony agents. So, all my agents behave honestly and follow the steps of the ceremony, but the human(s) might make mistakes. In other cases, such as in the SSO ceremony (see Section 5.8.5), I add an explicit attacker who intentionally tries to make the ceremony insecure.[4] In all these cases, my modelling of the human behaviour (through mutations to the specification of the human(s) and of the agents the human(s) interact with) allows us to identify attacks that a standard Dolev-Yao attacker would not immediately be able to find.

Σ also contains a countably infinite set $\mathcal{C}_{fresh}$ of fresh constants, modelling the generation of nonces, and a countably infinite set $\mathcal{C}_{pub}$ of public constants, representing agent names and other publicly known values. The sets $F_{sym}, \mathcal{C}_{fresh}$ and $\mathcal{C}_{pub}$ are pairwise disjoint. I denote sequences with square brackets.

[4] I control the Dolev-Yao attacker by using (or not) appropriate channels. The messages used in the Oyster ceremony are not encrypted, but there is no reason why they could not be. The SSO ceremony, in contrast, includes explicit cryptographic operations.

I say that m_1 is a *submessage* of m_2, in symbols $m_1 \in \mathit{submsg}(m_2)$, iff $m_2 = m_1$; $m_2 = \langle m_3, m_4 \rangle$ for some m_3, m_4 and $m_1 \in \mathit{submsg}(m_3)$ or $m_1 \in \mathit{submsg}(m_4)$; $m_2 = h(m_3)$ for some m_3 and $m_1 \in \mathit{submsg}(m_3)$; $m_2 = \mathit{senc}(m_3, k)$ for some m_3 and k and $m_1 \in \mathit{submsg}(m_3)$; $m_2 = \mathit{aenc}(m_3, k)$ for some m_3 and k and $m_1 \in \mathit{submsg}(m_3)$; or $m_2 = \mathit{sign}(m_3, k)$ for some m_3 and k and $m_1 \in \mathit{submsg}(m_3)$.

The *format* $f = \mathit{format}(m)$ of a message m is its top-level function symbol: if m has no top-level function symbol, then f is the identity function; if $m = \langle m_1, m_2 \rangle$ for some m_1, m_2, then $f = \mathit{pair}$; if $m = h(m_1)$ for some m_1, then $f = h$; if m is $\circ(m_1, k)$ for some m_1 and k, with $\circ \in \{\mathit{senc}, \mathit{aenc}, \mathit{sign}\}$, then $f = \circ$.

5.6.2 Ceremony Specification

Formally, a *role script* is a sequence of events $e \in \mathcal{T}_{\Sigma \cup \mathit{RoleActions}}(\mathcal{V})$, where $\mathit{RoleActions} = \{\mathit{Snd}, \mathit{Rcv}, \mathit{Start}, \mathit{Fresh}\}$ and each event e has exactly one function symbol that is in *RoleActions* at the top-level. I will introduce other Tamarin actions in my specifications (and simply write "actions" when there is no risk of confusion).

Send and receive events are of the form $\mathit{Snd}(A, l, P, m)$ and $\mathit{Rcv}(A, l, P, m)$, where A is the role executing the event, $l \in \mathit{LinkProp} = \{\mathit{ins}, \mathit{auth}, \mathit{conf}, \mathit{sec}\}$ indicates the type of channel over which a message is sent, $P \in \mathcal{C}_{pub}$ is a role's name, and $m \in \mathcal{T}_\Sigma(V)$ is a message. The channel types *ins*, *auth*, *conf* and *sec* denote insecure, authentic, confidential, and secure channels and correspond in the obvious manner to the channel symbols in the Alice&Bob notation (see [16] as well as [112, 113, 4] for a detailed discussion of different types of channels, including pseudonymous channels).

In the $\mathit{Snd}(A, l, P, m)$ event, P is the intended recipient of the message m, whereas in $\mathit{Rcv}(A, l, P, m)$ event, P is the apparent sender, as the attacker may have forged the message, and m is the expected message pattern.

Fresh(A, m) indicates that the role A generates a fresh message m (e.g., a nonce or a new key) and *Start*(A, K) indicates the initial knowledge K of A. The start event is the first event of a role script and occurs only once.

As shown in Figure 5.3b, the Generalised Main Ceremony for the Tube has 3 roles: the human H and the entrance and exit gates *GateIn* and *GateOut*. I remarked above that in this ceremony I do not consider cryptography (but I easily could) and, in fact, I do not consider an explicit attacker. I represent this by specifying that all messages are sent over secure channels. Thus, the role scripts for the roles of this ceremony are:

$$
\begin{aligned}
&\mathit{RoleScript}_H = \\
&\quad [\mathit{Start}(H, \langle \langle \text{`}\mathit{GateIn}\text{'}, \text{`}\mathit{GateOut}\text{'}, \text{`}\mathit{card}\text{'}\ \text{`}\mathit{balance}\text{'} \rangle \\
&\qquad\qquad \langle \mathit{GateIn}, \mathit{GateOut}, \mathit{card}, \mathit{bal}(\mathit{card}) \rangle \rangle), \\
&\quad \mathit{Snd}(H, \mathrm{sec}, \mathit{GateIn}, \langle \text{`}\mathit{card}\text{'}, \mathit{card} \rangle), \\
&\quad \mathit{Rcv}(H, \mathrm{sec}, \mathit{GateIn}, \langle \langle \text{`}\mathit{card}\text{'}, \text{`}\mathit{gin}\text{'} \rangle, \langle \mathit{card}, \mathit{gin} \rangle \rangle), \\
&\quad \mathit{Snd}(H, \mathrm{sec}, \mathit{GateOut}, \langle \langle \text{`}\mathit{card}\text{'}, \text{`}\mathit{balance}\text{'}, \text{`}\mathit{gin}\text{'} \rangle, \\
&\qquad\qquad \langle \mathit{card}, \mathit{bal}(\mathit{card}), \mathit{gin} \rangle \rangle), \\
&\quad \mathit{Rcv}(H, \mathrm{sec}, \mathit{GateOut}, \langle \langle \text{`}\mathit{card}\text{'}, \text{`}\mathit{balance}\text{'}, \text{`}\mathit{finish}\text{'} \rangle, \\
&\qquad\qquad \langle \mathit{card}, \mathit{bal}(\mathit{card})', \mathit{finish} \rangle \rangle)]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{RoleScript}_{\mathit{GateIn}} = \\
&\quad [\mathit{Start}(\mathit{GateIn}, \langle H, \mathit{gin} \rangle), \\
&\quad \mathit{Rcv}(\mathit{GateIn}, \mathrm{sec}, H, \langle \text{`}\mathit{card}\text{'}, \mathit{card} \rangle), \\
&\quad \mathit{Snd}(\mathit{GateIn}, \mathrm{sec}, H, \langle \langle \text{`}\mathit{card}\text{'}, \text{`}\mathit{gin}\text{'} \rangle, \langle \mathit{card}, \mathit{gin} \rangle \rangle)]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{RoleScript}_{\mathit{GateOut}} = \\
&\quad [\mathit{Start}(\mathit{GateOut}, \langle H, \mathit{gout}\rangle), \\
&\quad\ \mathit{Rcv}(\mathit{GateOut}, \mathrm{sec}, H, \langle\langle \text{`}\mathit{card}\text{'}, \text{`}\mathit{balance}\text{'}, \text{`}\mathit{gin}\text{'}\rangle, \\
&\qquad\qquad\qquad \langle \mathit{card}, \mathit{bal}(\mathit{card}), \mathit{gin}\rangle\rangle), \\
&\quad\ \mathit{Snd}(\mathit{GateOut}, \mathrm{sec}, H, \langle\langle \text{`}\mathit{card}\text{'}, \text{`}\mathit{balance}\text{'}, \text{`}\mathit{finish}\text{'}\rangle, \\
&\qquad\qquad\qquad \langle \mathit{card}, \mathit{bal}(\mathit{card})', \mathit{finish}\rangle\rangle)]
\end{aligned}
$$

I take advantage of *constants* in Tamarin to identify values received and sent during a ceremony. In [16], constants are used to define "tags" in order to represent the interpretation of the values in the knowledge of a human agent. I also make use of constants but I use them to define a basic notion of *types*. I only consider types of ground terms, such as the type '*card*' for *card* or '*balance*' for $\mathit{bal}(\mathit{oyster})$ as shown in the role scripts above and in the agent rules in Figure 5.8.[5] This allows us to restrict what mutations can do, e.g., constants allow us to express that a payment card in a message is replaced with another card (instead of with a generic value that is not of type "card").

Still, for readability,

> *from now I will often omit constants in role scripts and rules, so that when you read m, please mentally replace it with the constant-message pair $\langle t, m\rangle$.*

5.6.3 Execution Model

My approach is based on Tamarin's *execution model* [108], which is defined by a *multiset term-rewriting system* like in most other security protocol analysis tools. A *system state* is a multiset of *facts*: *linear facts* model exhaustible resources and they can be added to and removed from the system state, *persistent facts* model inexhaustible resources

[5] This is enough for all ceremonies that I have encountered so far, so I leave a more thorough investigation of types to future work.

$$[] \xrightarrow{Start(H,\langle GateIn,GateOut,oyster,balance\rangle)} [\mathsf{AgSt}(H, 1, \langle GateIn, GateOut, oyster, balance\rangle)] \quad (H_0)$$

$$[\mathsf{AgSt}(H, 1, \langle GateIn, GateOut, oyster, balance\rangle)] \xrightarrow{Snd(H,\mathrm{sec},GateIn,\langle `card', oyster\rangle)} [\mathsf{AgSt}(H, 2, \langle GateIn, GateOut, oyster, balance\rangle), \mathsf{Out}_{sec}(H, GateIn, \langle `card', oyster\rangle)] \quad (H_1)$$

$$[\mathsf{AgSt}(H, 2, \langle GateIn, GateOut, oyster, balance\rangle), \mathsf{In}_{sec}(GateIn, H, \langle\langle `card', `gin'\rangle, \langle oyster, gin\rangle\rangle)] \xrightarrow{Rcv(H,\mathrm{sec},GateIn,\langle\langle `card', `gin'\rangle,\langle oyster,gin\rangle\rangle)} [\mathsf{AgSt}(H, 3, \langle GateIn, GateOut, oyster, balance, gin\rangle)] \quad (H_2)$$

$$[\mathsf{AgSt}(H, 3, \langle GateIn, GateOut, oyster, balance, gin\rangle)] \xrightarrow{Snd(H,\mathrm{sec},GateOut,\langle\langle `card', `balance', `gin'\rangle,\langle oyster,bal(oyster),gin\rangle\rangle)} [\mathsf{AgSt}(H, 4, \langle GateIn, GateOut, oyster, balance, gin\rangle), \mathsf{Out}_{sec}(H, GateOut, \langle\langle `card', `balance', `gin'\rangle, \langle oyster, bal(oyster), gin\rangle\rangle)] \quad (H_3)$$

$$[\mathsf{AgSt}(H, 4, \langle GateIn, GateOut, oyster, balance, gin\rangle), \mathsf{In}_{sec}(GateOut, H, \langle\langle `card', `balance', `finish'\rangle, \langle oyster, bal(oyster)', finish\rangle\rangle)] \xrightarrow{Rcv(H,\mathrm{sec},GateOut,\langle\langle `card', `balance', `finish'\rangle,\langle oyster,bal(oyster)',finish\rangle\rangle),Hfin(H,`card',oyster)} [] \quad (H_4)$$

Figure 5.8 The rules for the human agent in the Generalised Main Ceremony for the Tube

and can only be added to the system state (persistent fact symbols are prefixed with "!"). The *initial system state* is the empty multiset. A *trace tr* is a finite sequence of multisets of actions a and is generated by the application of labelled *state transition rules* of the form

$$prem \xrightarrow{a} conc\,.$$

$$[] \xrightarrow{Start(GateIn,\langle H,gin\rangle)} [\mathsf{AgSt}(GateIn, 1, \langle H, gin\rangle)] \qquad (Gi_0)$$

$$[\mathsf{AgSt}(GateIn, 1, \langle H, gin\rangle), \mathsf{In}_{sec}(H, GateIn, \langle \text{`}card\text{'}, oyster\rangle)]$$
$$\xrightarrow{Rcv(GateIn,\mathrm{sec},H,\langle \text{`}card\text{'},oyster\rangle)} [\mathsf{AgSt}(GateIn, 2, \langle H, gin, oyster\rangle] \qquad (Gi_1)$$

$$[\mathsf{AgSt}(GateIn, 2, \langle H, gin, oyster\rangle]$$
$$\xrightarrow{Snd(GateIn,\mathrm{sec},H,\langle\langle \text{`}card\text{'},\text{`}gin\text{'}\rangle\langle oyster,gin\rangle\rangle),CommitGid(GateIn,H,gin)}$$
$$[\mathsf{Out}_{sec}(GateIn, H, \langle\langle \text{`}card\text{'}, \text{`}gin\text{'}\rangle, \langle oyster, gin\rangle\rangle)] \qquad (Gi_2)$$

Figure 5.9 Agent rules for the *GateIn* in the Generalised Main Ceremony for the Tube

$$[] \xrightarrow{Start(GateOut,\langle H,gout\rangle)} [\mathsf{AgSt}(GateOut, 1, \langle H, gout\rangle)] \qquad (Go_0)$$

$$[\mathsf{AgSt}(GateOut, 1, \langle H, gout\rangle),$$
$$\mathsf{In}_{sec}(H, GateOut, \langle\langle \text{`}card\text{'}, \text{`}balance\text{'}, \text{`}gin\text{'}\rangle, \langle oyster, bal(oyster), gin\rangle\rangle)]$$
$$\xrightarrow{Rcv(GateOut,\mathrm{sec},H,\langle\langle \text{`}card\text{'},\text{`}balance\text{'},\text{`}gin\text{'}\rangle,\langle oyster,bal(oyster),gin\rangle\rangle)}$$
$$[\mathsf{AgSt}(GateOut, 2, \langle H, GateOut, oyster, bal(oyster), gin\rangle] \qquad (Go_1)$$

$$[\mathsf{AgSt}(GateOut, 2, \langle H, GateOut, oyster, bal(oyster), gin\rangle]$$
$$\xrightarrow{\substack{Snd(GateOut,\mathrm{sec},H,\langle\langle \text{`}card\text{'},\text{`}balance\text{'},\text{`}finish\text{'}\rangle \\ \langle oyster,bal(oyster)',\text{`}finish\text{'}\rangle\rangle),Commit(GateOut,H,\text{`}finish\text{'})}}$$
$$[\mathsf{Out}_{sec}(GateOut, H, \langle\langle \text{`}card\text{'}, \text{`}balance\text{'}, \text{`}finish\text{'}\rangle,$$
$$\langle oyster, bal(oyster)', \text{`}finish\text{'}\rangle\rangle)] \qquad (Go_2)$$

Figure 5.10 Agent rules for the *GateOut* in the Generalised Main Ceremony for the Tube

Such a rule is applicable when the current state contains facts matching the premise *prem*, and the rule's application removes the matching linear facts from the state, adds instantiations of the facts in the conclusion *conc* to the state, and records the instantiations of actions in a in the trace. The set of all traces of a set of rules $\mathcal{R}$ is denoted by $TR(\mathcal{R})$.

For instance, the following denotes a trace of the actions of a human agent H and possibly pairwise distinct agents $A_1, A_2, A_3, A_4, \ldots$ where each Σ_i represents a possibly empty subtrace. When $A_1 = A_2 = \ldots = A$, trace (5.1) reduces to a trace of a ping-pong ceremony between H and a system A.

$$\begin{gathered}
\vdots\Sigma_1 \\
Rcv(H, l_1, A_1, m_1) \\
Snd(H, l_2, A_2, m_2) \\
Rcv(A_2, l_2, H, m_2) \\
\vdots\Sigma_2 \\
Rcv(H, l_3, A_3, m_3) \\
Snd(H, l_4, A_4, m_4) \\
Rcv(A_4, l_4, H, m_4) \\
\vdots\Sigma_3
\end{gathered} \tag{5.1}$$

A *protocol model* consists of the agent rules, the fresh rule, channel rules and attacker rules. The *fresh rule* $[\,] \rightarrow [\mathsf{Fr}(x)]$ produces the fact $\mathsf{Fr}(x)$ where $x \in \mathcal{C}_{fresh}$; no two applications of the fresh rule pick the same element $x \in \mathcal{C}_{fresh}$ and this is the only rule that can produce terms $x \in \mathcal{C}_{fresh}$. Tamarin comes equipped with standard Dolev-Yao *attacker rules* and with *channel rules* (introduced in [15]) to model the sending and receiving of messages over authentic/confidential/secure channels, and thus control the ability of the attacker (who, e.g., can not send, read or replay messages on a secure channel, although he might still be able to interrupt the communication).

Agent rules specify the agents' state transitions and communication. For instance, the rules for the human agent in the Generalised Main Ceremony for the Tube are shown in Figure 5.8 while the rules for the gatein agent in the Generalised Main Ceremony for the Tube are shown in Figure 5.9 and the rules for the gateout agent in the Generalised Main Ceremony for the Tube are shown in Figure 5.10. In general, for every event e in

the script of a role A, we get a transition rule $prem \xrightarrow{a} conc$ as follows: the label of the rule contains the event, i.e., $e \in a$; *prem* contains an agent state fact $\mathsf{AgSt}(A, step, kn)$, and *conc* contains the subsequent agent state fact $\mathsf{AgSt}(A, step, kn')$, where *step* refers to the role step the agent is in and *kn* is the agent's knowledge at that step. If $e \in a$ is: $Snd(A, l, P, m)$ then *conc* additionally contains an outgoing message fact $\mathsf{Out}_l(A, P, m)$; $Rcv(A, l, P, m)$ then *prem* contains an incoming message fact $\mathsf{In}_l(P, A, m)$; $Fresh(A, m)$ then *prem* contains $\mathsf{Fr}(m)$; $Start(A, m)$ then it is translated to a setup rule where *conc* contains the initial agent state $\mathsf{AgSt}(A, 0, m)$.[6]

As usual, the knowledge of an agent increases monotonically during the execution of the ceremony (as the agent receives messages or generates fresh terms).

5.6.4 Goals

Goals express the security properties that a ceremony is supposed to guarantee. However, many ceremonies, such as the Oyster ceremony as I discussed in Section 5.4, "push" security from the system to the human agents. This is made evident by the three goals that I define and analyse for the Oyster ceremony:

GO1 the human ends his journey touching in and out;

GO2 the human ends his journey using the same card to touch in and out;

GO3 the human does not touch two cards in and out.

These goals refer to a single journey, i.e., a single ceremony session. I formalise this in my Tamarin models by including explicit restrictions (through the *OnlyOnce* restriction [161] in the Setup phase; see my specifications in [178]) to force the human to carry out a single journey in the ceremony; given this, I can then formalise the

[6] The translation of the different channels into Tamarin is quite natural, e.g., by means of rules such as $\mathsf{Out}_l(A, P, m) \rightarrow \mathsf{Out}(A, P, m)$ for $l \in \{ins, auth\}$ and $\mathsf{In}(P, A, m) \rightarrow \mathsf{In}_l(P, A, m)$ for $l \in \{ins, conf\}$.

goals as follows. Goal *GO1* can be formalised by the lemma in Listing 5.1 which uses Tamarin *actions* to express that if H completes the ceremony with an Oyster card (action $Hfin(H, \text{`}card\text{'}, oyster)$) at time j, then there is a previous time instant i such that a *GateIn* commits the *gin* to H (action $CommitGid(GateIn, H, gin)$).

```
lemma complete journey: all-traces
  "All H oyster #j. Hfin(H,'card',oyster)@j ==>
   (Ex GateIn gin #i. CommitGid(GateIn,H,gin) @i & i<j)"
```

Listing 5.1 Lemma for the security goal *GO1*

The other goals make use of other actions. For instance, a $Snd(A, l, P, m)$ event corresponds to the Tamarin action $Send(A, \langle t, m \rangle)$, which I abbreviate to $Send(A, m)$ following my readability assumption, whereas a $Rcv(A, l, P, m)$ event corresponds to the action $Receive(A, P, m)$ (note the absence of the tag t).

Goal *GO2* can be formalised by the lemma in Listing 5.2 which expresses that if H completes the ceremony with an Oyster card ($Hfin(H, \text{`}card\text{'}, oyster)$) at time j, then H did not touch in another card.

```
lemma same card: all-traces
  "All H oyster #j. Hfin(H,'card',oyster)@j
   ==> (Ex #t. Send(H,'card',oyster)@t & t<j)
        & not (Ex ccard #c. Send(H,'card',ccard)@c
                          & not (ccard = oyster))"
```

Listing 5.2 Lemma for the security goal *GO2*

Goal *GO3* can be formalised by the lemma in Listing 5.3 which expresses that if *GateOut* receives *oyster* from H ($Receive(GateOut, H, oyster)$) and commits the end of the journey ($Commit(GateOut, H, \text{`}finish\text{'})$) at time j, and a *GateIn*, which is not instantiated by the same agent as *GateOut*, receives *oyster* from H and commits a *gin* ($CommitGid(GateIn, H, gin)$) to the same H at a previous t, then there does not exist another card *ccard* such that the *GateOut* and the *GateIn* execute the same transitions receiving that *ccard*.

```
lemma Card_Clash_Out: all-traces
  "All H GateIn GateOut oyster gin #j #t.
   Receive(GateOut,H,oyster)@j
   & Commit(GateOut,H,'finish')@j
   & Receive(GateIn,H,oyster)@t
   & CommitGid(GateIn,H,gin)@t &  t<j
   & not (GateIn = GateOut)
   ==> not (Ex ccard #i #k. Receive(GateOut,H,ccard)@i
   & Commit(GateOut,H,'finish')@i
   & Receive(GateIn,H,ccard)@k
   & CommitGid(GateIn,H,gin)@k & k<i
   & not oyster = ccard)"
```

Listing 5.3 Lemma for the security goal *GO3*

5.6.5 Threat Models

Oyster Using the approach formalised in Chapter 4, it is possible to highlight the following principals for the Oyster ceremony:

$$Humans(Oyster) := \{H\}$$
$$Technicals(Oyster) := \{GateIn, GateOut\}$$
$$Principals(Oyster) := Humans(Oyster) \ \cup \ Technicals(Oyster)$$

The most general case in which every principal gets all possible labels (cf. Section 4.4.4) sees:

$$\begin{aligned} n(PLHumans_{Oyster}(H)) &= 4 \\ n(PLTechnicals_{Oyster}(GateIn)) &= 4 \\ n(PLTechnicals_{Oyster}(GateOut)) &= 4 \end{aligned}$$

Hence, the full threat model chart has $4^3 = 64$ lines.

As highlighted in the motivation of this research, I have focussed on modelling the way humans make mistakes. This scenario is included as a result of one of the lines in the full threat model chart, as shown in Table 5.1.

	H	*GateIn*	*GateOut*
(a)	*error*	*normal*	*normal*

Table 5.1 The threat model considered in the Oyster ceremony

Some additional interesting cases can be found here. Scenarios where H acts choosing (*choice* in the charting) to do some actions are similar, in spirit, to what I envisaged (cf. Table 5.1) and are the motivations for this research (e.g., H can skip some actions to become an attacker and take advantage of the system). It would also be interesting to consider the scenario in Table 5.2, which captures an additional behavioural pattern where H faces a situation in which the gates are not working properly (e.g., H decides to call assistance, H decides to jump over, H waits, etc.) which I plan to investigate in future work.

	H	$GateIn$	$GateOut$
(b)	*choice*	*bug*	*bug*

Table 5.2 Additional threat model relevant for the Oyster ceremony

Single Sign-On For the Single Sign-On ceremony, it is possible to highlight the following principals:

$$Humans(SSO) := \{IdP\}$$
$$Technicals(SSO) := \{C, SP\}$$

I also allow for an attacking third party, so that the resulting principals are:

$$\begin{aligned} Principals(SSO) := \quad & Humans(SSO) \cup \\ & Technicals(SSO) \cup \\ & ATP(SSO) \end{aligned}$$

The most general case in which every principal gets all possible labels (cf. Section 4.4.4) sees:

$$\begin{aligned} n(PLHumans_{SSO}(IdP)) &= 4 \\ n(PLTechnicals_{SSO}(C)) &= 4 \\ n(PLTechnicals_{SSO}(SP)) &= 4 \\ n(PLATP_{SSO}) &= 2 \end{aligned}$$

Hence, the full threat model chart has $4^3 \cdot 2^1 = 128$ lines.

The charting here considers 128 lines, which result in many threat models. The well-known attack described in [7] was obtained by formalising Google's specific implementation (rather than the original specification) of the ceremony; instead, here I discover the attack by considering what would happen if *SP* is played by the attacker and *IdP* is played by a human who mistakenly sends a wrong message. This scenario

is included as a result of one of the lines in the full threat model chart, as shown in Table 5.3.

	C	*IdP*	*SP*	*ATP*
(a)	*normal*	*error*	*normal*	*malicious*

Table 5.3 The threat model considered in the Single Sign-On ceremony

However, my chart also highlights at least two more relevant threat models that haven't been considered yet and that are shown in Table 5.4.

	C	*IdP*	*SP*	*ATP*
(b)	*error*	*normal*	*normal*	—
(c)	*normal*	*normal*	*bug*	*malicious*

Table 5.4 Additional threat model relevant for the Single Sign-On ceremony

I believe that it will be interesting to consider also the threat models (b) and (c) as the threat model (b) suggests a scenario where C makes some mistakes, whereas the threat model (c) represents a relevant case, under C's point of view, in which SP is bugged and the system is under the attack of an ATP.

5.7 Modelling Human Mutations of the Ceremony

The mutations that humans carry out when executing a ceremony have repercussions also on other agents and thus on the whole ceremony. I thus need to define not only the human mutations, which modify a ceremony trace by mutating the subtrace of the human agent, but also the mutations on the subtrace(s) of the other agent(s) that are likely (albeit not necessarily) required to "match" the mutations of the human; for instance, to receive the new or modified message sent by the human or to skip some actions mirroring the skip of the human. The result will be a fully mutated ceremony

trace, which my tool feeds into Tamarin, first to check if it is executable and then to analyse it with respect to the corresponding goal(s).

Definition 1. *A generic* human mutation *is a function*

$$\mu^H : tr \to tr'$$

that takes as input a trace tr and gives as output a new trace $tr' = [\![tr]\!]^{\mu^H}$ *obtained by mutating H's subtrace as a consequence of the human H "deviating" from the original role script by skipping one or more actions, replacing a message with another one, adding a new action, or carrying out a human action that results in the removal of one or more Tamarin actions.*

A matching mutation *for a human mutation is a mutation* μ^m *that mutates the subtraces of the other ceremony agents to match and propagate the human mutation.*

The combination $\mu^H \circ \mu^m : tr \to tr'$ *of the two mutations takes as input a trace tr and gives as output a new trace* $tr' = [\![tr]\!]^{\mu^H \circ \mu^m}$ *in which the human mutation is matched and propagated.*

Note that in this part of the research, only "deviations" from the original role script by skipping one or more actions, replacing a message with another one and adding a new action are considered. A "deviation" that concerns a human action that results in the removal of one or more Tamarin actions will be taken into account in Chapter 6.

In the following subsections, I will instantiate these generic definitions to define the three human mutations *skip*, *replace* and *add* both formally and algorithmically, giving also the algorithmic definitions of the corresponding matching mutations. Slightly abusing notation, I will write $[a_0, \ldots, a_i, \ldots, a_n]^H$ with $0 < i \leqslant n$ to denote the subtrace of a human agent H in a ceremony execution, and let $[\![\_]\!]^\mu$ apply not just to traces. In fact, I consider mutations that apply generically to traces so that they apply indirectly also to role scripts and to Tamarin actions. For readability, and to make a clearer point,

in the following descriptions and algorithms, I will sometimes depart from the tight corset of Tamarin's notation and consider transitions and their pre and postconditions. More specifically, in the style of multiset rewriting as in [143], I consider an abstract "merged" transition rule in which *prem* contains the receipt of a message and *conc* the sending of the reply:[7]

$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow$$
$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ Snd(H, l_2, A_2, m_2)\,,$$

where Pre_i is a set of precondition facts (e.g., fresh facts) at state i, $Post_{i+1}$ is a set of postcondition facts at state $i+1$, and kn_{i+1} is obtained by extending kn_i with m_1 and with whatever is generated fresh in Pre_1. As usual, kn_{i+1} is such that A can send the message m_2 (after closing the knowledge under the standard rules for message generation and analysis). It is not difficult to translate this transition to the two corresponding transition rules in Tamarin's notation (with In, Out, the Tamarin actions and the constants) and vice versa, and to carry out the corresponding translations in the following descriptions and algorithms.

Then, the subtrace in (5.1) can be rewritten as follows, where I now embed A_2's receipt of m_2 in Σ_2 and A_4's receipt of m_4 in Σ_3 as I wish to focus on H's actions.

[7] This is in the spirit of the *step compression* technique that is adopted in several security protocol analysis tools, such as [5]. The idea is that some actions can be safely lumped together. For instance, I can safely assume that if a role is supposed to reply to a message it received, then I can compress the receive and send actions into a single transition.

$$
\begin{array}{c}
\vdots\Sigma_1 \\
\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow \\
\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ Snd(H, l_2, A_2, m_2) \\
\vdots\Sigma_2 \\
\mathsf{AgSt}(H, j, kn_j),\ Pre_j,\ Rcv(H, l_3, A_3, m_3) \rightarrow \\
\mathsf{AgSt}(H, j+1, kn_{j+1}),\ Post_{j+1},\ Snd(H, l_4, A_4, m_4) \\
\vdots\Sigma_3
\end{array}
\tag{5.2}
$$

5.7.1 The *skip* mutation

Definition 2. *A skip mutation*

$$\mu^H_{skip} : tr \rightarrow tr'$$

is a human mutation of tr's human subtrace $[a_0, \ldots, a_i, \ldots, a_n]^H$ *such that* tr' *includes the new human subtrace* $[a_0, \ldots, a_{i-1}, [\![a_{i+k}]\!]^\mu, \ldots, [\![a_n]\!]^\mu]$*, where* a_{i+k} *with* $k \geq 1$ *is the action that* H *executes immediately after the execution of* a_i *and* $[\![a_{i+k}]\!]^\mu, \ldots, [\![a_n]\!]^\mu$ *are the mutations of these actions obtained by* H *skipping the actions* $a_i, \ldots, a_{i+k-1}$ *and by matching and propagating this mutation.*

For example, Figure 5.11 shows a human subtrace in which H skips the $Snd(H, \mathrm{sec}, GateIn, card)$ action in the Oyster ceremony (omitting constants as discussed), which corresponds to not touching in. But this is not the only possible skip: H could skip also the receipt of the reply by *GateIn* and jump to his next send to *GateOut*, which would actually make sense as one could argue that if *GateIn* does not receive a message from H then it will not reply either; or H could skip both the receipt of a message and the sending of the reply; and so on.

$$
\begin{array}{l}
Start(H, \langle GateIn, GateOut, card, bal\rangle), \\
Snd(H, \text{sec}, GateIn, card), \\
Rcv(H, \text{sec}, GateIn, \langle card, gin\rangle), \\
Snd(H, \text{sec}, GateOut, \langle card, bal(card), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card, bal(card)', finish\rangle)
\end{array}
\xrightarrow{\mu^{H}_{skip(S)}}
\begin{array}{l}
Start(H, \langle GateIn, GateOut, card, bal\rangle), \\
\cancel{Snd(H, \text{sec}, GateIn, card)}, \\
Rcv(H, \text{sec}, GateIn, \langle card, gin\rangle), \\
Snd(H, \text{sec}, GateOut, \langle card, bal(card), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card, bal(card)', finish\rangle)
\end{array}
$$

$$
\begin{array}{l}
Start(H, \langle GateIn, GateOut, card, bal\rangle), \\
Snd(H, \text{sec}, GateIn, card), \\
Rcv(H, \text{sec}, GateIn, \langle card, gin\rangle), \\
Snd(H, \text{sec}, GateOut, \langle card2, bal(card2), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card2, bal(card2)', finish\rangle)
\end{array}
\xrightarrow{\mu^{H}_{replace}}
\begin{array}{l}
Start(H, \langle GateIn, GateOut, card, bal\rangle), \\
Snd(H, \text{sec}, GateIn, card), \\
Rcv(H, \text{sec}, GateIn, \langle card, gin\rangle), \\
Snd(H, \text{sec}, GateOut, \langle card2, bal(card2), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card2, bal(card2)', finish\rangle)
\end{array}
$$

$$
\begin{array}{l}
Start(H, \langle GateIn, GateOut, card, bal\rangle), \\
Snd(H, \text{sec}, GateIn, card), \\
Rcv(H, \text{sec}, GateIn, \langle card, gin\rangle), \\
Snd(H, \text{sec}, GateOut, \langle card, bal(card), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card, bal(card)', finish\rangle)
\end{array}
\xrightarrow{\mu^{H}_{add}, \mu^{H}_{replace}}
\begin{array}{l}
Snd(H, \text{sec}, GateOut, \langle card, bal(card), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card, bal(card)', finish\rangle) \\
\\
Snd(H, \text{sec}, GateOut, \langle card2, bal(card2), gin\rangle), \\
Rcv(H, \text{sec}, GateOut, \langle card2, bal(card2)', finish\rangle)
\end{array}
$$

Figure 5.11 Examples of mutations in the case of the Oyster ceremony

I have identified five different *skip* mutations, depending on which send (S) and receive (R) actions are skipped: $\mu^H_{skip(S)}$, $\mu^H_{skip(SR)}$, $\mu^H_{skip(R)}$, $\mu^H_{skip(RS)}$ and $\mu^H_{skip(RSR)}$. More cases could be considered, but these five cover the most interesting scenarios, which can combined to skip bigger "chunks" of the ceremony execution.

I describe the five *skip* mutations by showing their effect on the subtrace (5.2).

The *skip* mutation $\mu^H_{skip(S)}$

In this case, H, having arrived at state $i+1$, skips the sending of m_2 and any other action that he would carry out in Σ_2 and continues the trace with the transition $j \geq i+1$, which I call the *landing transition* (i.e., the transition where H lands after the "jump" he has made):[8]

$$\vdots\Sigma_1$$
$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow$$
$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ \cancel{Snd(H, l_2, A_2, m_2)}$$
$$\vdots[\![\Sigma_2]\!]^\mu$$
$$\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ Rcv(H, l_3, A_3, [\![m_3]\!]^\mu) \rightarrow$$
$$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$$
$$\vdots[\![\Sigma_3]\!]^\mu$$

where μ denotes the mutation composed of $\mu^H_{skip(S)}$ and the matching and propagation entailed by $\mu^H_{skip(S)}$ (mutations of constant-message pairs are explained later).

[8] For simplicity but w.l.o.g., in the following I assume that the (fresh and "other") facts in Pre_j never refer to messages received during the execution of a ceremony, but only to long-term keys, public keys and the like; this entails that $[\![Pre_j]\!]^\mu = Pre_j$. This assumption allows me to avoid considering mutations of Pre_j induced by the situation in which a message is not received in $[\![\Sigma_2]\!]^\mu$. This is indeed the case in the Oyster and SSO examples. Extending my approach to capturing such mutations is cumbersome notationally but not difficult technically: I can define the mutation of the preconditions (and of the postconditions, if needed) in a way similar to the mutation of the knowledge when one or more messages are not received.

This allows me to illustrate the need for matching and propagation on a concrete example. I namely need to consider if and how the mutated trace can be completed, for instance when H receives from A_3 an m_3 that is different from the expected one as a consequence of H's skipping the sending of m_2 to A_2. This immediately raises a number of questions. For instance, for $Rcv(H, l_3, A_3, [\![m_3]\!]^\mu)$ to be possible, it must be the case that $[\![\Sigma_2]\!]^\mu$ contains $Snd(A_3, l_3, H, [\![m_3]\!]^\mu)$, but there is no guarantee that this holds:

- if A_3 is able to send m_3 even when H does not send m_2 to A_2, then H can receive m_3, but
- if A_3 needs first A_2 (which is possibly but not necessarily equal to A_3) to receive m_2 to then be able to send m_3, then A_3 does not send m_3 in the mutated trace or sends a mutation of m_3 built from its current knowledge.

My tool implements these options as described in the pseudo-code in Algorithm 1 and Algorithm 2.

Algorithm 1 $\mu^H_{skip(S)}$: skip $Snd(H, l_2, A_2, m_2)$ in transition i, with landing transition j as in (5.2)

1: **if** $[\![kn_j]\!]^\mu = kn_j = kn_{i+1}$ **then**
2: **if** $[\![\Sigma_2]\!]^\mu$ still contains a transition with $Snd(A_3, l_3, H, m_3)$ in its conclusions **then**
3: transition j is the same as the original one in trace tr, i.e.
4: $\mathsf{AgSt}(H, j, kn_j)$, Pre_j, $Rcv(H, l_3, A_3, m_3) \rightarrow$ $\mathsf{AgSt}(H, j+1, kn_{j+1})$, $Post_{j+1}$, $Snd(H, l_4, A_4, m_4)$
5: **else** ▷ $[\![\Sigma_2]\!]^\mu$ does not contain a transition with $Snd(A_3, l_3, H, m_3)$ in its conclusions
6: build all transitions j for all mutations $[\![m_3]\!]^\mu = \{(format(m_3))(m) \mid m \in submsg(m_3)\}$ of m_3 and for each of these, set $[\![kn_{j+1}]\!]^\mu = kn_j \cup \{[\![m_3]\!]^\mu\} \cup Pre_j$ and build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{j+1}]\!]^\mu$, i.e.

7: $\mathsf{AgSt}(H, j, kn_j),\ Pre_j,\ Rcv(H, l_3, A_3, [\![m_3]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$

8: **else** ▷ this case is when $kn_j = kn_{i+1} \cup X$ for some set $X = \{M_1^x, ..., M_n^x\}$ of messages all different from m_3 and received by H in Σ_2

9: **if** $[\![\Sigma_2]\!]^\mu$ still contains a transition with $Snd(A_3, l_3, H, m_3)$ in its conclusions **then**

10: H skipped all transitions of Σ_2 in which he received $M_1^x, ..., M_n^x$,

11: $[\![kn_j]\!]^\mu = kn_j = kn_{i+1}$,

12: $[\![kn_{j+1}]\!]^\mu = kn_j \cup \{m_3\} \cup Pre_j$,

13: build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{j+1}]\!]^\mu$, i.e.

14: $\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ Rcv(H, l_3, A_3, m_3) \rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$

15: **else** ▷ $[\![\Sigma_2]\!]^\mu$ does not contain a transition with $Snd(A_3, l_3, H, m_3)$ in its conclusions

16: H skipped all transitions of Σ_2 in which he received $M_1^x, ..., M_n^x$ and cannot receive m_3 in its original form but only in its mutated form $[\![m_3]\!]^\mu$

17: $[\![kn_j]\!]^\mu = kn_j = kn_{i+1}$,

18: **go to** 6

The pseudo-code is hopefully quite explanatory, also thanks to the comments in the algorithms (whose start is denoted by ▷), but there are a couple of steps that deserve clarification. First of all, what does it mean that A_3 sends a mutation of m_3 built from its current knowledge? If I apply the message generation and analysis rules freely, this is an infinite set of possible messages. I could consider that as there is no guarantee of termination in my approach anyway, but instead I proceed in a more controlled way that mimics human users making mistakes when sending the messages or human programmers making mistakes when implementing a specification:

I consider only mutations of a message m that preserve the format of m.

So, for example, in line 6 of Algorithm 1 I define $[\![m_3]\!]^\mu = \{(format(m_3))(m) \mid m \in submsg(m_3)\}$ of m_3, and then, for each of these mutations, I build all the corresponding

transitions j (similarly, I build controlled mutations of m_4 in lines 6 and 13). Note also that I write $kn_l \cup Pre_l$ to mean the extension of kn_l with all messages generated freshly in Pre_l.

Algorithm 2 Matching mutation for $\mu^H_{skip(S)}$

1: Consider the transition $next(i)$ that immediately follows the mutated human transition i

2: $\mathsf{AgSt}(A_2, x, kn_x),\ Pre_x,\ Rcv(A_2, l_2, H, m_2) \rightarrow$
$\mathsf{AgSt}(A_2, x+1, kn_{x+1}),\ Post_{x+1},\ Snd(A_2, l_p, A_s, m_p)$ where A_s is one of the other agents and l_p and m_p are some channel and message as specified in Σ_2.

3: remove $Rcv(A_2, l_2, H, m_2)$ from $next(i)$

4: $[\![kn_x]\!]^\mu = kn_{x-1}$

5: $[\![kn_{x+1}]\!]^\mu = kn_x \cup Pre_x$

6: build all $[\![m_p]\!]^\mu = \{(format(m_p))(m) \mid m \in submsg(m_p)\}$ that can be generated by $[\![kn_{x+1}]\!]^\mu$ i.e.

7: $\mathsf{AgSt}(A_2, x, [\![kn_x]\!]^\mu),\ Pre_x \rightarrow$
$\mathsf{AgSt}(A_2, x+1, [\![kn_{x+1}]\!]^\mu),\ Post_{x+1},\ Snd(A_2, l_p, A_s, [\![m_p]\!]^\mu)$

8: Let $h ::= next(i)$

9: **if** $\exists\, next(h)$ i.e.

10: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y,\ Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},\ Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then**

11: $[\![kn_s]\!]^\mu = kn_{s-1}$

12: $m_p = [\![m_p]\!]^\mu$

13: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$

14: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.

15: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu),\ Pre_s,\ Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$

16: **go to** 9 **with** $h ::= next(h)$

The *skip* mutation $\mu^H_{skip(SR)}$

In this case, H skips $Snd(H, l_2, A_2, m_2)$ in transition i and $Rcv(H, l_3, A_3, m_3)$ in transition j:

$$\vdots\Sigma_1$$

$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow$$

$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ \cancel{Snd(H, l_2, A_2, m_2)}$$

$$\vdots[\![\Sigma_2]\!]^\mu$$

$$\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ \cancel{Rcv(H, l_3, A_3, m_3)} \rightarrow$$

$$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$$

$$\vdots[\![\Sigma_3]\!]^\mu$$

The pseudo-code for this case is in Algorithm 3 and in Algorithm 4.

Algorithm 3 $\mu^H_{skip(SR)}$: skip $Snd(H, l_2, A_2, m_2)$ in i and $Rcv(H, l_3, A_3, m_3)$ in j

1: **if** $[\![kn_j]\!]^\mu = kn_j = kn_{i+1}$ **then**
2: $\quad [\![kn_{j+1}]\!]^\mu = kn_j \cup Pre_j$,
3: $\quad$ build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{j+1}]\!]^\mu$, i.e.
4: $\quad \mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j, \rightarrow$
$\quad\quad \mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$
5: **else** $\quad \triangleright$ this case is when $kn_j = kn_{i+1} \cup X$ for some set $X = \{M^x_1, ..., M^x_n\}$ of messages all different from m_3 and received by H in Σ_2
6: $\quad H$ skipped all transitions of Σ_2 in which he received $M^x_1, ..., M^x_n$,
7: $\quad [\![kn_j]\!]^\mu = kn_j = kn_{i+1}$,
8: $\quad$ **go to** 1

Algorithm 4 Matching mutation for $\mu^H_{skip(SR)}$

1: Consider the transition $next(i)$ that immediately follows the mutated human transition i
2: $\mathsf{AgSt}(A_2, x, kn_x),\ Pre_x,\ Rcv(A_2, l_2, H, m_2) \rightarrow$

$\mathsf{AgSt}(A_2, x+1, kn_{x+1})$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, m_p)$ where A_s is one of the other agents and l_p and m_p are some channel and message as specified in Σ_2.

3: remove $Rcv(A_2, l_2, H, m_2)$ from $next(i)$

4: $[\![kn_x]\!]^\mu = kn_{x-1}$

5: $[\![kn_{x+1}]\!]^\mu = kn_x \cup Pre_x$

6: build all $[\![m_p]\!]^\mu = \{(format(m_p))(m) \mid m \in submsg(m_p)\}$ that can be generated by $[\![kn_{x+1}]\!]^\mu$ i.e.

7: $\mathsf{AgSt}(A_2, x, [\![kn_x]\!]^\mu)$, $Pre_x \rightarrow$
$\mathsf{AgSt}(A_2, x+1, [\![kn_{x+1}]\!]^\mu)$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, [\![m_p]\!]^\mu)$

8: Let $h ::= next(i)$

9: **if** $\exists\, next(h)$ i.e.

10: $\mathsf{AgSt}(A_s, s, kn_s)$, Pre_y, $Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1})$, $Post_{s+1}$, $Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then**

11: **if** s is different than j **then**

12: $[\![kn_s]\!]^\mu = kn_{s-1}$

13: $m_p = [\![m_p]\!]^\mu$

14: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$

15: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.

16: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu)$, Pre_s, $Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu)$, $Post_{s+1}$,
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$

17: **go to** 9 **with** $h ::= next(h)$

18: **else** ▷ s is equal to j, so this is the transition j in which I have to remove the Rcv

19: remove $Rcv(A_s, l_p, A_{s-1}, m_p)$ from transition j

20: $\mathsf{AgSt}(A_s, s, kn_s)$, Pre_y, $\rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1})$, $Post_{s+1}$, $Snd(A_s, l_p, A_{s+1}, m_{p+1})$ is in the form

21: $\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu)$, Pre_j, $\rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu)$, $Post_{j+1}$, $Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$

22: Let $h ::= next(j)$

23: **if** $\exists\, next(h)$ i.e.

24: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y,\ Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, m_{p+1})$

then

25: $[\![kn_s]\!]^\mu = kn_{s-1}$

26: $m_p = [\![m_p]\!]^\mu$

27: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$

28: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.

29: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu),\ Pre_s,\ Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$

30: **go to** 23 **with** $h ::= next(h)$

The *skip* mutation $\mu^H_{skip(R)}$

H skips $Rcv(H, l_1, A_1, m_1)$ in transition i:

$$\vdots\Sigma_1$$

$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ \cancel{Rcv(H, l_1, A_1, m_1)} \rightarrow$$

$$\mathsf{AgSt}(H, i+1, [\![kn_{i+1}]\!]^\mu),\ Post_{i+1},\ Snd(H, l_2, A_2, [\![m_2]\!]^\mu)$$

$$\vdots[\![\Sigma_2]\!]^\mu$$

$$\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ Rcv(H, l_3, A_3, m_3) \rightarrow$$

$$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$$

$$\vdots[\![\Sigma_3]\!]^\mu$$

The pseudo-code for this case is in Algorithm 5 and in Algorithm 6.

Algorithm 5 $\mu^H_{skip(R)}$: skip $Rcv(H, l_1, A_1, m_1)$ in transition i

1: **if** $[\![kn_{i+1}]\!]^\mu = kn_{i+1} = kn_i \cup Pre_j$ **then**

2: transition i is the same as the original one in trace t, without the $Rcv(H, l_1, A_1, m_1)$, i.e.

3: $\mathsf{AgSt}(H, i, kn_i),\ Pre_i \rightarrow$

$\mathsf{AgSt}(H, i+1, kn_{i+1})$, $Post_{i+1}$, $Snd(H, l_2, A_2, m_2)$

4: **else** ▷ this case is when $kn_{i+1} = kn_i \cup X \cup Pre_i$, that means that m_1 has new knowledge

5: $[\![kn_{i+1}]\!]^{\mu} = kn_i \cup Pre_i$,

6: build all $[\![m_2]\!]^{\mu} = \{(format(m_2))(m) \mid m \in submsg(m_2)\}$ that can be generated by $[\![kn_{i+1}]\!]^{\mu}$, i.e.

7: $\mathsf{AgSt}(H, i, kn_i)$, $Pre_i \rightarrow$
$\mathsf{AgSt}(A, i+1, [\![kn_{i+1}]\!]^{\mu})$, $Post_{i+1}$, $Snd(H, l_2, A_2, [\![m_2]\!]^{\mu})$

Algorithm 6 Matching mutation for $\mu^{H}_{skip(R)}$

1: Consider the transition $next(i)$ that immediately follows the mutated human transition i

2: $\mathsf{AgSt}(A_2, x, kn_x)$, Pre_x, $Rcv(A_2, l_2, H, m_2) \rightarrow$
$\mathsf{AgSt}(A_2, x+1, kn_{x+1})$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, m_p)$ where A_s is one of the other agents and l_p and m_p are some channel and message as specified in Σ_2.

3: **if** m_2 in $Snd(H, l_2, A_2, m_2)$ in transition i is sent without modification **then**

4: $\mathsf{AgSt}(A_2, x, kn_x)$, Pre_x, $Rcv(A_2, l_2, H, m_2) \rightarrow$
$\mathsf{AgSt}(A_2, x+1, kn_{x+1})$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, m_p)$

5: Let $h ::= next(i)$

6: **if** $\exists\, next(h)$ i.e.

7: $\mathsf{AgSt}(A_s, s, kn_s)$, Pre_y, $Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1})$, $Post_{s+1}$, $Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then** ▷ no modification are necessary to the next transition

8: $\mathsf{AgSt}(A_s, s, kn_s)$, Pre_y, $Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1})$, $Post_{s+1}$, $Snd(A_s, l_p, A_{s+1}, m_{p+1})$

9: **go to** 6 **with** $h ::= next(h)$

10: **else** ▷ this case is when $[\![m_2]\!]^{\mu}$ is sent (modifications have been applied)

11: $\mathsf{AgSt}(A_2, x, kn_x)$, Pre_x, $Rcv(A_2, l_2, H, [\![m_2]\!]^{\mu}) \rightarrow$
$\mathsf{AgSt}(A_2, x+1, [\![kn_{x+1}]\!]^{\mu})$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, [\![m_p]\!]^{\mu})$

12: Let $h ::= next(i)$

13: **if** $\exists\, next(h)$ i.e.

14: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y,\ Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},\ Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then**
15: $[\![kn_s]\!]^\mu = kn_{s-1}$
16: $m_p = [\![m_p]\!]^\mu$
17: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$
18: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.
19: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu),\ Pre_s,\ Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$
20: **go to** 13 **with** $h ::= next(h)$

The *skip* mutation $\mu^H_{skip(RS)}$

H skips both $Rcv(H, l_1, A_1, m_1)$ and $Snd(H, l_2, A_2, m_2)$ in transition i:

$$\vdots\Sigma_1$$
$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ \cancel{Rcv(H, l_1, A_1, m_1)} \rightarrow$$
$$\mathsf{AgSt}(H, i+1, [\![kn_{i+1}]\!]^\mu),\ Post_{i+1},\ \cancel{Snd(H, l_2, A_2, [\![m_2]\!]^\mu)}$$
$$\vdots[\![\Sigma_2]\!]^\mu$$
$$\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ Rcv(H, l_3, A_3, m_3) \rightarrow$$
$$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$$
$$\vdots[\![\Sigma_3]\!]^\mu$$

The pseudo-code for this case is in Algorithm 7 and in Algorithm 8.

Algorithm 7 $\mu^H_{skip(RS)}$: skip $Rcv(H, l_1, A_1, m_1)$ and $Snd(H, A_2, l_2, m_2)$ in transition i, with landing transition j

1: **if** $[\![kn_j]\!]^\mu = kn_j = kn_i$ **then**
2: **if** $[\![\Sigma_2]\!]^\mu$ still contains a transition with $Snd(A_3, l_3, H, m_3)$ in its RHS **then**
3: transition j is the same as the original one in trace t, i.e.
4: $\mathsf{AgSt}(H, j, kn_j),\ Pre_j,\ Rcv(H, l_3, A_3, m_3) \rightarrow$
$\mathsf{AgSt}(H, j+1, kn_{j+1}),\ Post_{j+1},\ Snd(H, l_4, A_4, m_4)$

5: **else** ▷ $[\![\Sigma_2]\!]^\mu$ does not contain a transition with $Snd(A_3, l_3, H, m_3)$ in its RHS

6: build all transitions j for all mutations $[\![m_3]\!]^\mu = \{(format(m_3))(m) \mid m \in submsg(m_3)\}$ of m_3 and for each of these, set $[\![kn_{j+1}]\!]^\mu = kn_j \cup \{[\![m_3]\!]^\mu\} \cup Pre_j$ and build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{j+1}]\!]^\mu$, i.e.

7: $[\![kn_{j+1}]\!]^\mu = kn_j \cup \{[\![m_3]\!]^\mu\} \cup Pre_j$,

8: build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{j+1}]\!]^\mu$, i.e.

9: $\mathsf{AgSt}(H, j, kn_j),\ Pre_j,\ Rcv(H, l_3, A_3, [\![m_3]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$

10: **else**▷ this case is when $kn_j = kn_i \cup X$ for some set X of messages all different from m_3 and received by H in Σ_2 with $X = \{M_1^x, ..., M_n^x\}$

11: **if** $[\![\Sigma_2]\!]^\mu$ still contains a transition with $Snd(A_3, l_3, H, m_3)$ in its RHS **then**

12: H skipped all transitions of Σ_2 in which he received $M_1^x, ..., M_n^x$,

13: $[\![kn_j]\!]^\mu = kn_j = kn_i$,

14: $[\![kn_{j+1}]\!]^\mu = kn_j \cup \{m_3\} \cup Pre_j$,

15: build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[[kn_{j+1}]\!]^\mu$, i.e.

16: $\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ Rcv(H, l_3, A_3, m_3) \rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$

17: **else** ▷ $[\![\Sigma_2]\!]^\mu$ does not contain a transition with $Snd(A_3, l_3, H, m_3)$ in its RHS

18: H skipped all transitions of Σ_2 in which he received $M_1^x, ..., M_n^x$ and he cannot receive m_3 in its original form but only in its mutation $[\![m_3]\!]^\mu$

19: $[\![kn_j]\!]^\mu = kn_j = kn_i$,

20: **go to** 6

Algorithm 8 Matching mutation for $\mu^H_{skip(RS)}$

1: Consider the transition $next(i)$ that immediately follows the mutated human transition i

2: $\mathsf{AgSt}(A_2, x, kn_x),\ Pre_x,\ Rcv(A_2, l_2, H, m_2) \rightarrow$

$\mathsf{AgSt}(A_2, x+1, kn_{x+1})$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, m_p)$ where A_s is one of the other agents and l_p and m_p are some channel and message as specified in Σ_2.

3: remove $Rcv(A_2, l_2, H, m_2)$ from $next(i)$

4: $[\![kn_x]\!]^\mu = kn_{x-1}$

5: $[\![kn_{x+1}]\!]^\mu = kn_x \cup Pre_x$

6: build all $[\![m_p]\!]^\mu = \{(format(m_p))(m) \mid m \in submsg(m_p)\}$ that can be generated by $[\![kn_{x+1}]\!]^\mu$ i.e.

7: $\mathsf{AgSt}(A_2, x, [\![kn_x]\!]^\mu)$, $Pre_x \rightarrow$
$\mathsf{AgSt}(A_2, x+1, [\![kn_{x+1}]\!]^\mu)$, $Post_{x+1}$, $Snd(A_2, l_p, A_s, [\![m_p]\!]^\mu)$

8: Let $h ::= next(i)$

9: **if** $\exists\, next(h)$ i.e.

10: $\mathsf{AgSt}(A_s, s, kn_s)$, Pre_y, $Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1})$, $Post_{s+1}$, $Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then**

11: $[\![kn_s]\!]^\mu = kn_{s-1}$

12: $m_p = [\![m_p]\!]^\mu$

13: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$

14: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.

15: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu)$, Pre_s, $Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu)$, $Post_{s+1}$,
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$

16: **go to** 9 **with** $h ::= next(h)$

The *skip* mutation $\mu^H_{skip(RSR)}$

H skips both $Rcv(H, l_1, A_1, m_1)$ and $Snd(H, l_2, A_2, m_2)$ in transition i and $Rcv(H, l_3, A_3, m_3)$ in transition j:

$$\vdots\Sigma_1$$

$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ \cancel{Rcv(H, l_1, A_1, m_1)} \rightarrow$$

$$\mathsf{AgSt}(H, i+1, [\![kn_{i+1}]\!]^\mu),\ Post_{i+1},\ \cancel{Snd(H, l_2, A_2, [\![m_2]\!]^\mu)}$$

$$\vdots[\![\Sigma_2]\!]^\mu$$

$$\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j,\ \cancel{Rcv(H, l_3, A_3, m_3)} \rightarrow$$

$$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$$

$$\vdots[\![\Sigma_3]\!]^\mu$$

The pseudo-code for this case is in Algorithm 9 and in Algorithm 10.

Algorithm 9 $\mu^H_{skip(RSR)}$: skip $Rcv(H, l_1, A_1, m_1)$ and $Snd(H, A_2, l_2, m_2)$ in i and $Rcv(H, l_3, A_3, m_3)$ in j

1: **if** $[\![kn_j]\!]^\mu = kn_j = kn_i$ **then**
2: $[\![kn_{j+1}]\!]^\mu = kn_j \cup Pre_j$,
3: build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{j+1}]\!]^\mu$, i.e.
4: $\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j \rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$
5: **else**▷ this case is when $kn_j = kn_i \cup X$ for some set X of messages all different from m_3 and received by H in Σ_2 with $X = \{m^x_1, ..., m^x_n\}$
6: H skipped all transitions of Σ_2 in which he received $m^x_1, ..., m^x_n$,
7: $[\![kn_j]\!]^\mu = kn_j = kn_i$,
8: **go to** 1

Algorithm 10 Matching mutation for $\mu^H_{skip(RSR)}$

1: Consider the transition $next(i)$ that immediately follows the mutated human transition i
2: $\mathsf{AgSt}(A_2, x, kn_x),\ Pre_x,\ Rcv(A_2, l_2, H, m_2) \rightarrow$
$\mathsf{AgSt}(A_2, x+1, kn_{x+1}),\ Post_{x+1},\ Snd(A_2, l_p, A_s, m_p)$ where A_s is one of the other agents and l_p and m_p are some channel and message as specified in Σ_2.

3: remove $Rcv(A_2, l_2, H, m_2)$ from $next(i)$
4: $[\![kn_x]\!]^\mu = kn_{x-1}$
5: $[\![kn_{x+1}]\!]^\mu = kn_x \cup Pre_x$
6: build all $[\![m_p]\!]^\mu = \{(format(m_p))(m) \mid m \in submsg(m_p)\}$ that can be generated by $[\![kn_{x+1}]\!]^\mu$ i.e.
7: $\mathsf{AgSt}(A_2, x, [\![kn_x]\!]^\mu),\ Pre_x \rightarrow$
$\mathsf{AgSt}(A_2, x+1, [\![kn_{x+1}]\!]^\mu),\ Post_{x+1},\ Snd(A_2, l_p, A_s, [\![m_p]\!]^\mu)$
8: Let $h ::= next(i)$
9: **if** $\exists\, next(h)$ i.e.
10: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y,\ Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},\ Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then**
11: **if** s is different than j **then**
12: $[\![kn_s]\!]^\mu = kn_{s-1}$
13: $m_p = [\![m_p]\!]^\mu$
14: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$
15: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.
16: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu),\ Pre_s,\ Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$
17: **go to** 9 **with** $h ::= next(h)$
18: **else** ▷ s is equal to j, so this is the transition j in which I have to remove the Rcv
19: remove $Rcv(A_s, l_p, A_{s-1}, m_p)$ from transition j
20: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y, \rightarrow$
21: $\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},\ Snd(A_s, l_p, A_{s+1}, m_{p+1})$ is in the form
22: $\mathsf{AgSt}(H, j, [\![kn_j]\!]^\mu),\ Pre_j, \rightarrow$
$\mathsf{AgSt}(H, j+1, [\![kn_{j+1}]\!]^\mu),\ Post_{j+1},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu)$
23: Let $h ::= next(j)$
24: **if** $\exists\, next(h)$ i.e.
25: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y,\ Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, m_{p+1})$

26: **then**
27: $[\![kn_s]\!]^\mu = kn_{s-1}$
28: $m_p = [\![m_p]\!]^\mu$
29: $[\![kn_{s+1}]\!]^\mu = [\![kn_s]\!]^\mu \cup Pre_s \cup [\![m_p]\!]^\mu$
30: build all $[\![m_{p+1}]\!]^\mu = \{(format(m_{p+1}))(m) \mid m \in submsg(m_{p+1})\}$ that can be generated by $[\![kn_{s+1}]\!]^\mu$ i.e.
31: $\mathsf{AgSt}(A_s, s, [\![kn_s]\!]^\mu),\ Pre_s,\ Rcv(A_s, l_p, A_{s-1}, [\![m_p]\!]^\mu) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, [\![kn_{s+1}]\!]^\mu),\ Post_{s+1},$
$Snd(A_s, l_p, A_{s+1}, [\![m_{p+1}]\!]^\mu)$
32: **go to** 24 **with** $h ::= next(h)$

5.7.2 The *replace* mutation

This mutation captures the fact that a human user may send a message in place of another one (the case in which H replaces an action of a ceremony with another one is obtained by combining a *skip* and an *add* mutation).

If I was to allow the human to replace a message to send with any message that he can build out of his current knowledge, then I would have to deal with an infinite set of options (even if the human knows only one thing, such as his name, the message generation and analysis rules will allow him to generate an infinite set of messages); I will have the same problem with the *add* mutation that I consider next. In security protocol analysis, the ability of the attacker to generate infinitely many actors is a cause of non-termination of the analysis, which is controlled by considering only the messages that honest agents will actually respond to (and by introducing symbolic techniques, such as the "lazy intruder" [113], to manage the remaining infinite set of "answerable" messages). I cannot do that as my approach generates mutations in the behaviour of the other agents to match the human mutations, so the other agents will be able to respond to any human message. Similar to [16], I thus restrict my attention to the messages that are already in the human's current knowledge *kn* (rather than in the knowledge's closure under the generation and analysis rules). More specifically, I

consider $\mathcal{P}(kn) \setminus \emptyset$, the *powerset* of kn, i.e., the finite set of all subsets of kn including kn but excluding the empty set as it does not make sense to send an empty message. Moreover, to simplify further, I restrict my attention to messages of the same type. I leave the investigation of other controlled notions of "sendable" messages to future work.

Definition 3. *A replace mutation*

$$\mu^H_{replace} : tr \to tr'$$

is a human mutation of tr's human subtrace $[a_0, \ldots, a_i, \ldots, a_n]^H$ *such that* tr' *includes the new human subtrace* $[a_0, \ldots, [\![a_i]\!]^\mu, [\![a_{i+1}]\!]^\mu, \ldots, [\![a_n]\!]^\mu]$*, where* a_i *is a send action* $Snd(H, l, A, m)$ *and* $[\![a_i]\!]^\mu$ *is its mutation obtained by replacing the message* m *either with a sub-message (but preserving the format) or with a message contained in the powerset* $\mathcal{P}(kn_i) \setminus \emptyset$ *of* H*'s current knowledge (but preserving types as specified by the corresponding constants);* $[\![a_{i+1}]\!]^\mu, \ldots, [\![a_n]\!]^\mu$ *are the mutations of the actions* $a_{i+1}, \ldots, a_n$ *obtained by matching and propagating this mutation.*

Again, I show the effect of the $\mu^H_{replace}$ mutation by showing its effect on the subtrace (5.2):

$$\begin{gathered}
\vdots \Sigma_1 \\
\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \to \\
\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ Snd(H, l_2, A_2, [\![m_2]\!]^\mu) \\
\vdots [\![\Sigma_2]\!]^\mu \\
\mathsf{AgSt}(H, i+1, [\![kn_{i+1}]\!]^\mu),\ Pre_{i+1},\ Rcv(H, l_3, A_3, [\![m_3]\!]^\mu) \to \\
\mathsf{AgSt}(H, i+2, [\![kn_{i+2}]\!]^\mu),\ Post_{i+2},\ Snd(H, l_4, A_4, [\![m_4]\!]^\mu) \\
\vdots [\![\Sigma_3]\!]^\mu
\end{gathered}$$

where μ, which denotes the mutation composed of $\mu^H_{replace}$ and the matching and propagation entailed by $\mu^H_{replace}$, replaces m_2 either with $[\![m_2]\!]^\mu = \{(format(m_2))(m) \mid m \in submsg(m_2)\}$ or with a message m that has the same constant as m_2 and is obtained from H's current knowledge as shown in Algorithm 11 along with Algorithm 12 for the matching mutation.[9]

For example, H could start the Oyster ceremony with one card *card*1 and complete it with another card *card*2, thus giving rise to two "incomplete journeys", as shown in Figure 5.11 and discussed in Section 5.8.4. For another example, see the SSO ceremony in Section 5.8.5.

Algorithm 11 *replace* mutation $\mu^H_{replace}$

1: build all transitions i obtained by replacing m_2 either with each $[\![m_2]\!]^\mu = \{(format(m_2))(m) \mid m \in submsg(m_2)\}$ or with each $[\![m_2]\!]^\mu$ that is in the powerset of H's current knowledge kn_{i+1} preserving types as specified by the corresponding constants, i.e.

2: $\mathsf{AgSt}(H, i, kn_i),\ Pre_i, Rcv(H, l_1, A_1, m_1) \rightarrow$
$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ Snd(H, l_2, A_2, [\![m_2]\!]^\mu)$ ▷ Σ_2 does not contain other transitions by H

3: **if** $[\![\Sigma_2]\!]^\mu$ still contains a transition with $Snd(A_3, l_3, H, m_3)$ in its conclusions **then** ▷ the new message $[\![m_2]\!]^\mu$ has no influence on m_3

4: $\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Pre_{i+1}, Rcv(H, l_3, A_3, m_3) \rightarrow$
$\mathsf{AgSt}(H, i+2, kn_{i+2}),\ Post_{i+2},\ Snd(H, l_4, A_4, m_4)$

5: **else**

6: **if** $[\![\Sigma_2]\!]^\mu$ contains a transition with $Snd(A_3, l_3, H, [\![m_3]\!]^\mu)$ in its conclusions **then** ▷ the new message $[\![m_2]\!]^\mu$ has some influence on m_3

7: $[\![kn_{i+2}]\!]^\mu = kn_{i+1} \cup [\![m_3]\!]^\mu \cup Pre_{i+1}$,

8: build all $[\![m_4]\!]^\mu = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{i+2}]\!]^\mu$, as defined in 1 i.e.

9: $\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Pre_{i+1}, Rcv(H, l_3, A_3, [\![m_3]\!]^\mu) \rightarrow$

[9] I give only one algorithm for $\mu^H_{replace}$ since, differently from the cases of the *skip* mutations, the two subcases of $\mu^H_{replace}$ proceed in the same way after the replacement of m_2.

$\mathsf{AgSt}(H, i+2, [\![kn_{i+2}]\!]^{\mu}),\ Post_{i+2},\ Snd(H, l_4, A_4, [\![m_4]\!]^{\mu})$

10: **else** ▷ $[\![\Sigma_2]\!]^{\mu}$ does not contain a transition with $Snd(A_3, l_3, H, m_3)$ or $Snd(A_3, l_3, H, [\![m_3]\!]^{\mu})$ in its conclusions (the new message $[\![m_2]\!]^{\mu}$ blocks the sending of m_3)

11: $[\![kn_{i+2}]\!]^{\mu} = kn_{i+1} \cup Pre_{i+1}$,

12: build all $[\![m_4]\!]^{\mu} = \{(format(m_4))(m) \mid m \in submsg(m_4)\}$ that can be generated by $[\![kn_{i+2}]\!]^{\mu}$, as defined in 1 i.e.

13: $\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Pre_{i+1} \rightarrow$
$\mathsf{AgSt}(H, i+2, [\![kn_{i+2}]\!]^{\mu}),\ Post_{i+2},\ Snd(H, l_4, A_4, [\![m_4]\!]^{\mu})$

Algorithm 12 Matching mutation for $\mu^{H}_{replace}$

1: Consider the transition $next(i)$ that immediately follows the mutated human transition i

2: $\mathsf{AgSt}(A_2, x, kn_x),\ Pre_x,\ Rcv(A_2, l_2, H, m_2) \rightarrow$
$\mathsf{AgSt}(A_2, x+1, kn_{x+1}),\ Post_{x+1},\ Snd(A_2, l_p, A_s, m_p)$ where m_2 could be either $[\![m_2]\!]^{\mu} = \{(format(m_2))(m) \mid m \in submsg(m_2)\}$ or $[\![m_2]\!]^{\mu}$ that is in the powerset of H's current knowledge kn_{i+1} preserving types as specified by the corresponding constants, A_s is one of the other agents and l_p and m_p are some channel and message as specified in Σ_2.

3: **if** $[\![m_2]\!]^{\mu} = \{(format(m_2))(m) \mid m \in submsg(m_2)\}$ **then**

4: $[\![kn_{x+1}]\!]^{\mu} = kn_x \cup Pre_x \cup [\![m_2]\!]^{\mu}$

5: $[\![m_p]\!]^{\mu} = m_p$ after removing all the messages that are not in $[\![kn_{x+1}]\!]^{\mu}$.

6: **else** ▷ $[\![m_2]\!]^{\mu}$ is in the powerset of H's current knowledge kn_{i+1} preserving types as specified by the corresponding constants

7: $[\![m_p]\!]^{\mu} = m_p$ after changing all the messages preserving types as specified by the corresponding constants.

8: Let $h ::= next(i)$

9: **if** $\exists\, next(h)$ i.e.

10: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_y,\ Rcv(A_s, l_p, A_{s-1}, m_p) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1}),\ Post_{s+1},\ Snd(A_s, l_p, A_{s+1}, m_{p+1})$ **then**

11: $[\![kn_s]\!]^{\mu} = kn_{s-1}$

12: $m_p = [\![m_p]\!]^{\mu}$

13: $[\![kn_{s+1}]\!]^{\mu} = [\![kn_s]\!]^{\mu} \cup Pre_s \cup [\![m_p]\!]^{\mu}$
14: **if** $[\![m_p]\!]^{\mu} = \{(format(m_{p-1}))(m) \mid m \in submsg(m_{p-1})\}$ **then**
15: $[\![m_{p+1}]\!]^{\mu} = m_{p+1}$ after removing all the messages that are not in $[\![kn_{s+1}]\!]^{\mu}$.
16: **else** ▷ $[\![m_p]\!]^{\mu}$ is generated using H's current knowledge kn_{p-1}
17: $[\![m_{p+1}]\!]^{\mu} = m_{p+1}$ after changing all the messages preserving types as specified by the corresponding constants.
18: **go to** 9 **with** $h ::= next(h)$

5.7.3 The *add* mutation

There are two different cases for this mutation: the human could

- send at any time any message that is in the powerset of the messages that are in his current knowledge, or
- duplicate a send action.[10]

Definition 4. *An add mutation is a mutation*

$$\mu^{H}_{add} : tr \to tr \times tr'$$

such that the original trace $tr = [a_0, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n]$ *is run in parallel with the new, mutated trace* $tr' = [a_0, \ldots, a_{i-1}, [\![a_i]\!]^{\mu}, [\![a_{i+1}]\!]^{\mu}, \ldots, [\![a_n]\!]^{\mu}]$*, where* a_i *is a send action and* $[\![a_i]\!]^{\mu}$ *is its possible mutation obtained either by duplicating* a_i *or by adding an action* $Snd(H, l, A, m)$ *at state* i *for some* l *with* $A, m \in \mathcal{P}(kn_i) \setminus \emptyset$*, and* $[\![a_{i+1}]\!]^{\mu}, \ldots, [\![a_n]\!]^{\mu}$ *are the mutations of the actions* $a_i, \ldots, a_{i+k-1}$ *obtained by matching and propagating this mutation.*

Consider the beginning of the subtrace (5.2). μ^{H}_{add} mutates this to either

[10]Note that I only consider mutations initiated by a human agent; as a consequence, I do not consider the situation in which the human agent initiates a mutation of the ceremony by adding a receive action as that would require another agent (human or not) to have added the corresponding send action first.

$$\vdots\Sigma_1$$
$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, M_1) \rightarrow$$
$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ Snd(H, l_2, A_2, M_2)$$
$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, M_1) \rightarrow$$
$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Snd(H, l, A, M)$$
$$\vdots[\![\Sigma_2]\!]^{\mu}$$

or

$$\vdots\Sigma_1$$
$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, M_1) \rightarrow$$
$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Post_{i+1},\ Snd(H, l_2, A_2, M_2)$$
$$\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, M_1) \rightarrow$$
$$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Snd(H, l_2, A_2, [\![M_2]\!]^{\mu})$$
$$\vdots[\![\Sigma_2]\!]^{\mu}$$

The add mutation is best exemplified when combined with the other mutations, e.g.,

- sending at any time any message that he knows can be combined with a *skip* mutation (e.g., to skip some steps of a ceremony and instead send an arbitrary message before continuing with the rest of the ceremony),
- duplicating a send action can be combined with a *replace* mutation (as I do in my case studies).

For example, H could start the Oyster ceremony touching in with one card *card*1 and then, by mistake, touch out with two cards; this can be represented by adding a second touch-out send message where the first card is replaced with the second, thus obtaining the two traces shown in Figure 5.11.

My tool implements this mutation as described in the pseudo-code in Algorithm 13 along with Algorithm 14 for the matching mutation.

Algorithm 13 *add* mutation μ^H_{add}

1: Add a transition at state i built by either
2: adding a $Snd(H, l, A, m)$ for some l, A and $m \in \mathcal{P}(kn_i) \setminus \emptyset$ preserving types as specified by the corresponding constants, where Pre_i contains only fresh facts (namely those fresh messages needed to built m; hence Pre_i could be empty), keeping the premises fixed as the same as the state i, i.e.
3: $\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow$
$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Snd(H, l, A, m)$
4: or duplicating an existing $Snd(H, l, A, m_2)$ action, keeping the premises fixed as the same as the state i, i.e.
5: $\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow$
$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Snd(H, l, A, m_2)$

Algorithm 14 Matching mutation for μ^H_{add}

1: Consider the new mutated human transition i
2: $\mathsf{AgSt}(H, i, kn_i),\ Pre_i,\ Rcv(H, l_1, A_1, m_1) \rightarrow$
$\mathsf{AgSt}(H, i+1, kn_{i+1}),\ Snd(H, l, A_s, m)$, where A_s is one of the other agents and l and m are some channel and message as specified in Algorithm 13.
3: Considering the receiver A_s, take its $next(i)$ transition that immediately follows the mutated human transition i.
4: Create a copy $\overline{next(i)}$ of the $next(i)$ transition and in $\overline{next(i)}$ remove the $Snd()$ event (if any) and replace the values in the $Rcv()$ event with l, m and A_s
5: $\mathsf{AgSt}(A_s, s, kn_s),\ Pre_s,\ Rcv(A_s, l, H, m) \rightarrow$
$\mathsf{AgSt}(A_s, s+1, kn_{s+1})$
6: For all transitions x after the transition $next(i)$
7: $\mathsf{AgSt}(A, x, kn_x) \ldots \rightarrow \mathsf{AgSt}(H, x+1, kn_{x+1}) \ldots$
8: increment the state increasing the role step the agent is in but keeping the rest of the transitions intact.

5.8 X-Men: a Tool for the Generation of Mutated Models based on Human Behaviours

In this section, I show how my formalisation can be used effectively for discovering attacks that are due to the behaviour of human agents in security ceremonies. As proof-of-concept, I have applied my tool X-Men to two case studies, the Oyster ceremony and the Single Sign-On ceremony. As explained in more detail in the following, X-Men generated a large number of mutated models for these ceremonies, which I have analysed using Tamarin.

The framework works in three phases as shown in Figure 5.1:

- A preprocessing phase executed by a Python script;
- The core phase executed by X-Men;
- The analysis phase executed by Tamarin.

5.8.1 Tamarin Model of a Ceremony

Ceremony models follow the Tamarin syntax. However, the X-Men framework needs to take as input *model files* that follow some extra expedients.

.spthy file

A Tamarin specification (a specification file in the .spthy format, where .spthy stands for security protocol theory) is constituted by the construct described in Section 2.3. Tamarin uses C-style comments, so everything between `/*` and `*/` or the line following `//` is a comment. I take advantage of comments in the X-Men framework to divide the internal structure of the models in three sections as shown in Listing 5.4.

```
/****MODEL****/

theory theory_name

begin

/* Channel rules */

/****RULES****/

builtins: ...

functions: ...

/* Protocol/Ceremony Rules */

/****ENDOFRULES****/

/* Restrictions and Lemmas */

end

/****ENDOFMODEL****/
```

Listing 5.4 Structure of the Tamarin models for the X-Men framework

The comments `/****MODEL****/` and `/****ENDOFMODEL****/` set the boundaries of the entire file. Within these boundaries, there are:

- an inner bound (from `/****MODEL****/` to `/****RULES****/`) where the channel rules are defined;
- an inner bound (from `/****RULES****/` to `/****ENDOFRULES****/`) where functions and the ceremony rules are defined;
- an inner bound (from `/****ENDOFRULES****/` to `/****ENDOFMODEL****/`) where the restrictions and the lemmas are defined.

Channel Rules

Channel rules need to be specified as follows: the fact names for outgoing and incoming message fact have to initiate with the string `Snd` and `Rcv` (e.g., `SndS()` for a secure outbound communication, `RcvS()` for a secure incoming communication, `SndDY()` for an insecure outbound communication, `RcvDY()` for an insecure incoming communication). The X-Men tool is able to parse channel rules as specified in Section 5.6.3, where the rules can have three or four parameters based on the presence of "types" as shown in Listing 5.5 (the full channel rules used in the ceremonies are in the Oyster model in Section A.3 and in the Single Sign-On model in Section A.4).

```
rule ChanSndS:
  [SndS($A,$B,xn,x)]
  --[ChanSndS($A,$B,xn,x)]->
  [!Sec($A,$B,xn,x)]


rule ChanRcvS:
  [!Sec($A,$B,xn,x)]
```

```
--[ChanRcvS($A,$B,xn,x)]->
[RcvS($A,$B,xn,x)]
```

Listing 5.5 Example of channel rules defined in Tamarin for the Oyster Ceremony

Human Agents

The rules executed by an human agent can be identified using the fact `H()` as first fact defined in the action part of each rule of the human agent.

```
rule rule_name:
                [ ... ]
                --[ H()
                , ...]->
                [ ... ]
```

Setup

In order to process the the roles inside a ceremony, X-Men needs them to be specified inside the actions of a rule named `Setup`. In protocol and ceremony analysis, we can usually take advantage of a setup rule to define the initial states of each role, however, in this case, the tool requires that a fact `Roles(...)` is instantiated. The arguments of this fact should be the agent names of the roles used in the protocol or ceremony as shown for the Oyster ceremony in Listing 5.6, where `Human`, `GateIn` and `GateOut` are respectively the agent names of the human passenger H, the gatein *GateIn* and the gateout *GateOut*.

```
rule Setup:
    [ ... ]
    --[ ...
    , Roles($Human,$GateIn,$GateOut)
    ...
    ]->
    [ ...]
```

Listing 5.6 Example of the Setup rule for the Oyster Ceremony

Additionally, if "types" are required, a rule named `humansetup` needs to be defined, containing persistent facts of the form `!Type(...)` in its postconditions. As shown in Listing 5.7, the syntax of a `!Type(...)` rule has:

- the agent name as first parameter;
- the type as second parameter;
- the variable that has that specific type as third parameter.

```
rule humansetup:
  [ ... ]
  --[ ... ]->
  [ ...
    !Type($Human,'card',$oyster)
    , !Type($Human,'card',$ccard)
    , !Type($Human,'balance',bal($oyster))
    , !Type($Human,'balance',bal($ccard))
```

```
    , !Type($GateIn,'gid',~gid)
    , !Type($GateIn,'goid',~goid)
    ...
    ]
```

Listing 5.7 Example of the humansetup rule for the Oyster Ceremony

Rules

Except for the `Setup`, the `humansetup` and the channel rules, the rules have to be defined (and written) in the specification following their actual execution in the protocol/ceremony. Considering the Generalised Main Ceremony for the Tube in Figure 5.3, the order of the rules should be as in Listing 5.8 (cf. the Oyster rules in Section A.3, the Single Sign-On rules in Section A.4 and the rules Section A.5 used in Chapter 6 follow the same criteria).

```
rule H_1:
        [...]--[...]->[...]
rule GateIn_1:
        [...]--[...]->[...]
rule H_3:
        [...]--[...]->[...]
rule GateOut_1:
        [...]--[...]->[...]
rule H_5:
        [...]--[...]->[...]
```

Listing 5.8 The order of the rules defined for the Oyster Ceremony

5.8.2 The Python Script: *Wolverine*

Wolverine is the Python script that deals with splitting a ceremony model, written in Tamarin, in three parts. It takes as input a model of the security ceremony *filename.spthy* with the boundaries described above and creates as output three new files:

(i) a new file that contains channel rules, named *filename_pre.spthy*,

(ii) a new file that contains the functions and all the ceremony agent rules, named *filename.spthy*,

(iii) a new file that contains restrictions and lemmas, named *filename_post.spthy*.

In addition to these files, another file named *filename.spthy_tmp* is created to preserve the original model file with a backup copy. Once the single-file model has been divided, Wolverine invokes the X-Men tool opening its GUI and waiting, in background, for the next interaction with the framework.

5.8.3 X-Men: the Core of the X-Men Framework

Under the Hood of X-Men

X-Men is a prototype tool written in Java. The tool follows the *Model–view–controller (MVC)* paradigm dividing the related program logic into the three interconnected elements of the model, a view (or GUI) and the controller which interconnects the model and the view. X-Men model includes two key components represented by an ANTLR grammar [123], essential to make the tool read the syntax of the files, and by a parser that elaborates the file based on the ANTLR grammar in order to create a structure easy to manipulate. The grammar is written following the syntax of the Tamarin grammar in [161].

Using X-Men

The security analyst, once the files have been created, should see the X-Men GUI as in Figure 5.12.

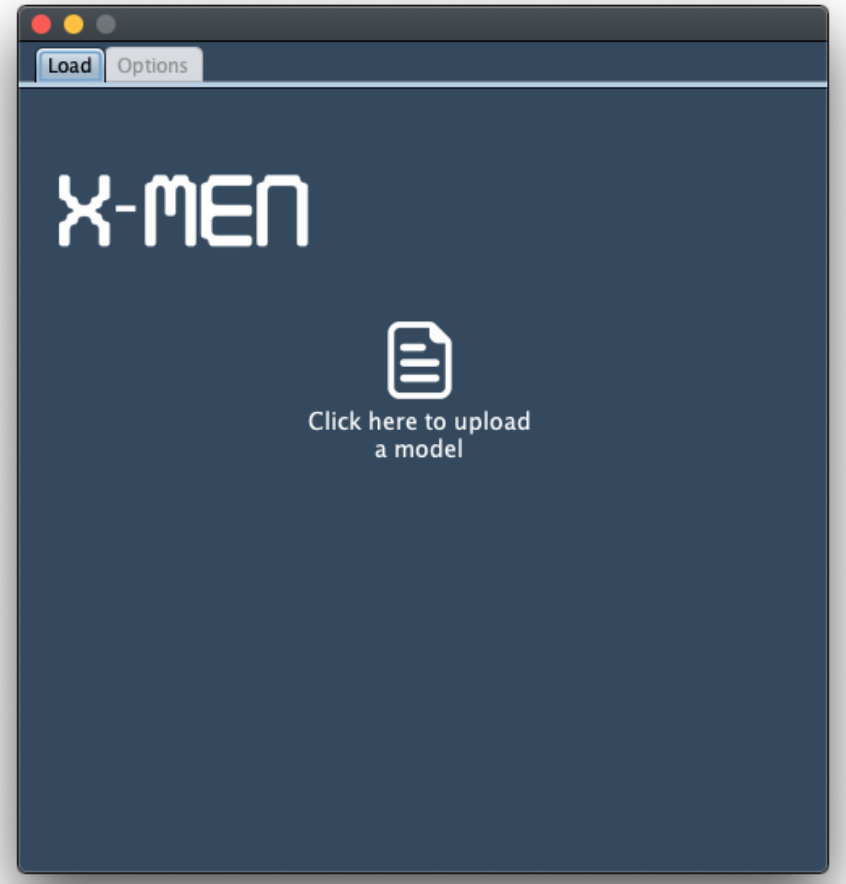


Figure 5.12 The GUI of X-Men

The security analyst employing X-Men then selects the desired mutation as shown in Figure 5.13. The analyst can choose the three mutations I have defined here or their combinations (once the single mutations are selected, the related combinations are shown in the GUI in the designate area) and any other mutation that will be defined

in X-Men's library of behavioural patterns in the future, to mutate the agent and the other rules.

Once X-Men has finished the generation of the mutations, then *Wolverine* merges the mutated agent and the other rules with the original channel rules and goals to produce the many different mutated models that can be input to Tamarin.

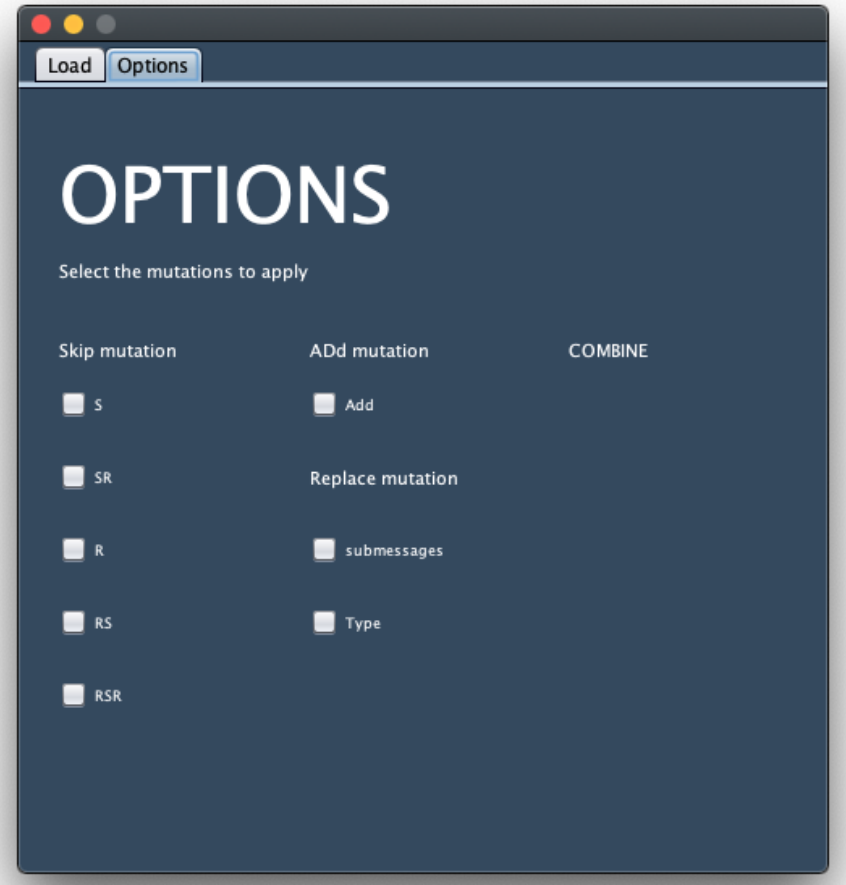


Figure 5.13 The mutations implemented in X-Men

X-Men generated a large number of mutated models for my two case studies, as shown in Table 5.5.

These mutated models were then analysed individually by Tamarin, activating the attacker rules when necessary (i.e., for the SSO case study but not for the Oyster one). Let me now summarize the results of the analysis.

Mutation		**Attacker**	**Case Study**	
			Oyster	SSO
skip	S	-	2	-
	SR	-	3	-
	R	-	2	-
	RS	-	1	-
	RSR	-	1	-
replace	submessage	✓	-	232
	type	-	2	-
add		-	92	
add&replace	submessage	✓	-	232
	type	-	2	-

Table 5.5 Mutated models generated by X-Men

5.8.4 Analysis of the Oyster Ceremony

Table 5.6 shows some of the attacks found with the models obtained using the mutations applied to the Oyster ceremony. The table does not show the results pertaining to the 92 models generated by the *add* mutation. These results can be safely omitted here as they represent cases in which the human agent systematically tries all the possible combinations of parameters, without following the logic behind the ceremony. In these cases, in fact, the security analyst needs to analyse each file excluding those who do not represent a plausible Oyster ceremony.

Mutation	Mutated model	Mutation details	Explanation	Goal: complete journey (*GO1*)	Goal: same card (*GO2*)	Goal: card clash (*GO3*)	Goal: functional
skip	S M0	Skip send in H_1	H does not touch in	✓	✓	×	•
	M1	Skip send in H_2	H does not touch out	×	×	×	•
	SR M0	Skip send in H_1 and receive in H_2	H does not touch in (here H ignores any response from *GateIn*)	✓	×	×	•
	M1	Skip send in H_1 and receive in H_3	H does not touch in (here H could receive a response from *GateIn* but ignores any response from *GateOut*)	✓	✓	×	•
	M2	Skip send in H_2 and receive in H_3	H does not touch out (here H ignores any response from *GateOut*)	×	×	×	•
	R M0	Skip receive in H_2	H does not receive confirmation of touch in	✓	×	×	•
	M1	Skip receive in H_3	H does not receive confirmation of touch out	×	×	×	•
	RS M0	Skip receive and send in H_2 and receive in H_3	H does not receive confirmation of touch in and does not touch out (here H could receive a response from *GateOut*)	×	×	×	•

Table 5.6 Some of the attacks found on the models obtained using the mutations applied to the Oyster ceremony
(✓ indicates that an attack has been found, × indicates that no attack was found, • indicates that the functional goal is verified)

Mutation	Mutated model	Mutation details	Explanation	Goal: complete journey (*GO1*)	Goal: same card (*GO2*)	Goal: card clash (*GO3*)	Goal: functional
	RSR	M0	Skip receive and send in H_2 and receive in H_3	✓	×	×	•
replace	M0	Replace *oyster* with *ccard* in whole ceremony	H uses a contactless card instead of Oyster	×	×	×	•
	M1	Replace *oyster* with *ccard* only in H_2 and after	H touches out using a different card	×	✓	×	•
	M2	Replace *bal*(*oyster*) with *bal*(*ccard*)	H uses balance of ccard instead of Oyster	×	×	×	•
add&replace	M0	Similar to "replace M0" keeping the original ceremony	H uses a contactless card instead of Oyster	×	✓	✓	•
	M1	Similar to "replace M1" keeping the original ceremony	H touches out using a different card	×	✓	×	•
	M2	Similar to "replace M2" keeping the original ceremony	H uses balance of ccard instead of Oyster	×	×	×	•

Table 5.6 (Continued) Some of the attacks found on the models obtained using the mutations applied to the Oyster ceremony
(✓ indicates that an attack has been found, × indicates that no attack was found, • indicates that the functional goal is verified)

"Mutated model" lists the file identifier of the generated file (as used at [178]) and the table also provides mutation details as well as a brief explanation of the human agent's behaviour for each model. In addition to the three goals discussed in Section 5.6.4, I have used Tamarin to check the *functional* goal (a.k.a. *executable* goal) that the mutations did not create models in which the legitimate execution trace of the ceremony is not valid any more. All the models considered in the table passed this check.

I describe three interesting attacks that Tamarin has been able to discover out of the many mutations generated.

Attack #1 The MSC of the attack (Figure 5.14) shows how the human agent H may execute the Oyster ceremony without touching-in at the entrance (as shown by the dotted arrows representing the human agent who is not touching-in). H touches-out the *oyster* and *GateOut* reads the information saved on the card, which does not specify where H entered as *GateIn* was not able to write its identifier *gin* on the card. The security goal *GO1* is not verified, entailing what TfL calls an *incomplete journey* as mentioned in Section 5.4, and the system charges a penalty fare as it is not able to calculate the journey of the passenger.

This is a real scenario that occurs when the passenger forgets to touch-in, e.g., when the station has no proper gates but only card readers at the station entrance, when the gates are already open (TfL sometimes opens the gates to speed up entry/exit during rush hour or when there are a large number of passengers), or when the reader is not working properly and does not read/write the Oyster card.

Attack #2 H may use two different cards in a single journey, touching-in with the first and touching-out with the second, so that *GO2* fails with two incomplete journeys. This may appear to be an uncommon scenario, but several tourists and even Londoners

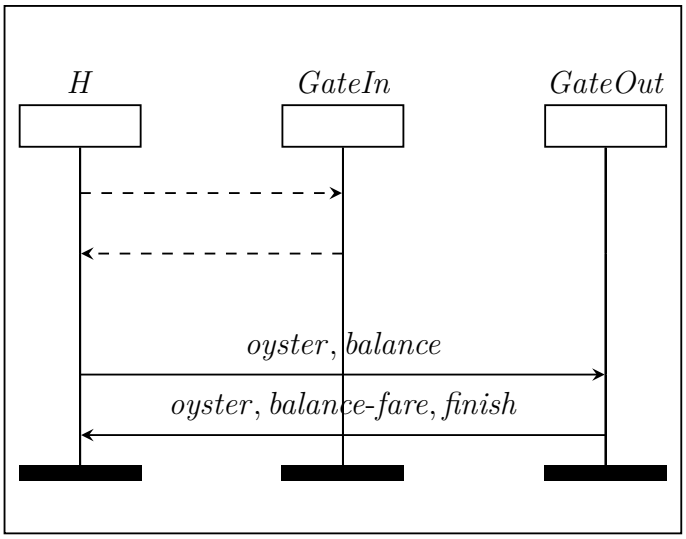


Figure 5.14 Attack that represents the Incomplete journey scenario for the Oyster ceremony

suffered from this problem. For instance, the passenger might have two Oyster cards in their pockets and confuse them, or the passenger might use Apple/Google pay (cf. Section 5.4) but using two different devices (say smartphone and smartwatch), which will cause two incomplete journeys because the *Device Account Number* is unique for each device and is used by TfL as the identifier for a single journey.

Attack #3 The MSC in Figure 5.15 shows how H may use two cards (e.g., Oyster and a contactless card), simultaneously touching them both in/out when entering/exiting (as shown by the dotted arrows representing a parallel second execution of the Oyster ceremony), so that *GO3* fails due to a *card clash* (cf. Section 5.4). This occurs, e.g., when a passenger touches with a wallet that holds all the passenger's cards that the system considers to be valid payment cards.

The attacks on the Oyster ceremony were discovered using the mutations generated by X-Men as shown in Table 5.6. The analysis did not require the activation of a Dolev-Yao attacker as the system, through the matching mutations, replied and billed the passengers "wrongly" due to their mistakes. While these attacks are, to some extent,

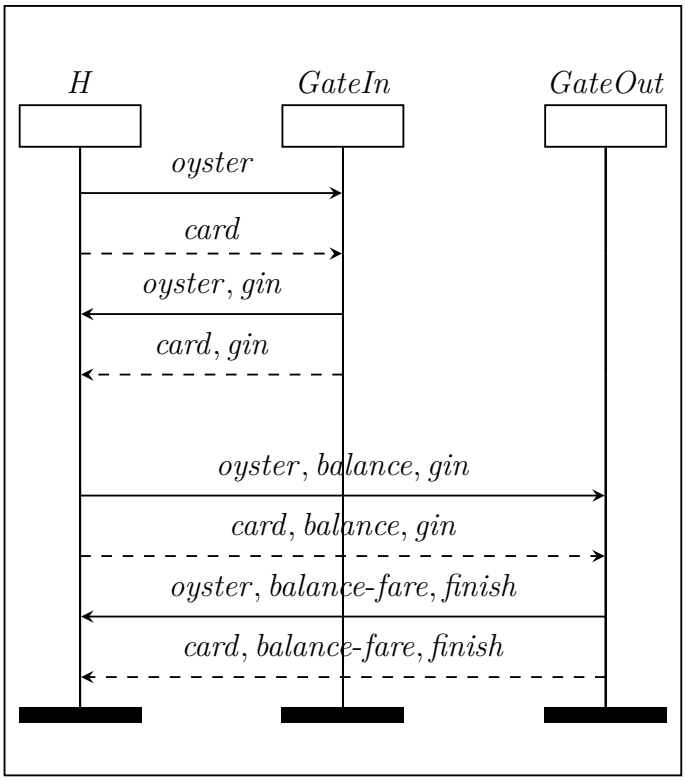


Figure 5.15 Attack that represents the Card clash scenario for the Oyster ceremony

known to TfL (cf. their warnings in Figure 5.4) and can be gathered empirically by observing the concrete behaviour of the Tube passengers, it is important to stress that X-Men allows us to discover them automatically. Other attacks might be discovered by considering other goals or other mutations. Moreover, in the style of model-based testing (see the end of Section 5.5), it is possible to use the attack traces to generate concrete test cases to be executed on the code of the ceremony (if that is available).

To provide an example of the analysis in the presence of an attacker, I have considered the SSO ceremony.

5.8.5 Analysis of the SSO Ceremony

The *SAML-based Single Sign-on for Google Apps* protocol suffered from a well-know man-in-the-middle attack [7] as shown in Figure 5.16. The protocol relies on the use of

an authentication assertion $AuthAssert(ID, C, IdP, SP)$ signed by the identity provider IdP, where ID is an identifier and SP is the provider of a service that a client C wishes to use. In a nutshell (see [7] for the full details), the attack was due to the fact that the implementation had simplified $AuthAssert(ID, C, IdP, SP)$ into $AuthAssert(C, SP)$ to speed up the digital signature. A malicious SP can use this assertion to pose as C in another run of the protocol.

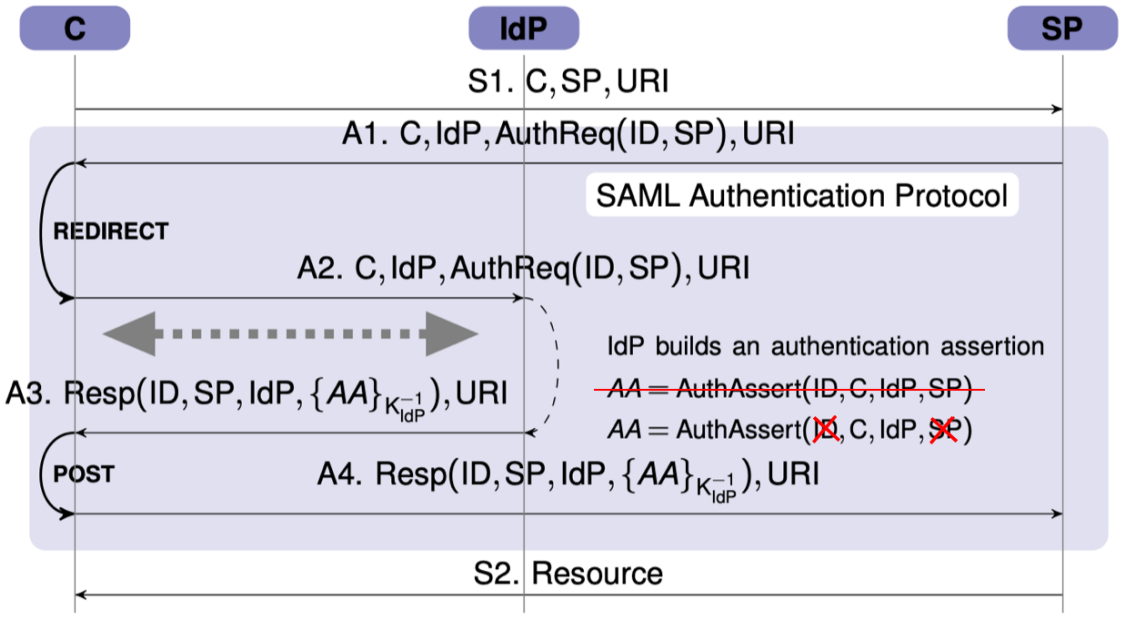


Figure 5.16 The man-in-the-middle attack on the Single Sign-On protocol

I have specified SSO as a ceremony in X-Men, considering what would happen if SP is played by the attacker and IdP is played by a human, who may mistakenly generate and sign a wrong authentication assertion. Indeed, the *replace (submessage)* mutation generates $AuthAssert(C, SP)$ among other mutations. I have formalised the goal "IdP authenticates only the agent who requires to be authenticated" as a standard injective-agreement goal in Tamarin as in Listing 5.9 and indeed I was able to find the attack.

```
lemma injective_agree:
   "All actor peer params #i.
        Commit(actor, peer, params)@ i
        ==>
         (Ex #j. Running(actor, peer, params)@ j & j < i
          & not(Ex actor2 peer2 #i2.
                Commit(actor2, peer2, params)@ i2
                & not(#i = #i2)))
        | (Ex #r. RevLtk(actor) @ r)
        | (Ex #r. RevLtk(peer)  @ r)"
```

Listing 5.9 Lemma for the security goal of SSO

This shows that my approach is able to find an attack that was not present in the original specification of SAML-based Single Sign-on but was introduced in Google's implementation [7]. My mutations, among other things, capture such possible specification-implementation deviances.

5.8.6 The X-Men Framework - Reusability

As said in Section 5.6, X-Men extends the Tamarin prover, but my approach is general and independent of Tamarin and could be applied similarly to other tools. To consider the use of other tools rather than the Tamarin prover, it is necessary to adapt some parts of the X-Men framework.

In Tamarin (and thus in the X-Men tool), a role is formalised by a so-called role script, which is basically the projection to an individual role of an extended Alice&Bob specification, and corresponds to a strand or an applied pi calculus process.

The *ceremony specification* is expressed using role scripts. Adapting the ceremony specification to a new tool is straightforward if the tool is using strands of applied pi calculus processes.

As Tamarin's *execution model* is defined by a *multiset term-rewriting system* like in most other security protocol analysis tools, it should require minor changes in case the new tool uses the same execution model (e.g., AVANTSSAR [5] uses the ASLan++ language which is defined by a *multiset term-rewriting system* as well).

Security goals are expressed using *lemmas* as LTL formulas. Changes to adapt the syntax of LTL operators are required to adapt the goals from one tool to another. In particular, it is necessary to exclude, in some scenarios, the Dolev-Yao attacker, allowing the goals to be analyzed even without it. This would potentially require an in-depth modification of the model checker code in order to exclude the attacker rules.

As described in Section 5.8.3, X-Men works using the MVC paradigm. Inside the model, there are two key components, which are an ANTLR grammar and a parser that elaborates the file based on the ANTLR grammar. The ANTLR grammar is based on the Tamarin syntax, which allows X-Men to read the .spthy files. So, to make X-Men able to read new syntax, it would be necessary to import a new grammar expressed using ANTLR, generating the parser and connecting it to the main components of the X-Men controller which, indipendently from the language, create the structure easy to manipulate in order to generate the mutations.

5.9 Related Work

In [124], Paulson introduced the *Oops* rule to model mistakes done by agents when executing a security protocol, such as the loss (by any means) of a session key. However, the notion of security ceremony and the explicit investigation of the consequences

of explicitly considering human agents and their mistakes was introduced by Ellison in [58], one of the pioneers of *socio-technical security*.

One of the first formal approaches to investigate security ceremonies is the *concertina* model introduced in [20], which spans over a number of socio-technical layers, focusing in particular on the socio-technical protocol between a user persona and a computer interface, but without explicitly considering human mistakes nor accounting for an explicit attacker. Similarly, the approaches in [103, 36] provide a formal model to reason about how a Dolev-Yao-style attacker can attack the communication between humans and computers, including storing of human knowledge, but without explicitly considering human mistakes.

In contrast, Basin et al. [16] provide a formal model for reasoning about some errors that humans involved in security protocols may make. They specify rules formalising different types of humans (untrained, infallible or fallible humans), modelling a human who can send and receive any messages, resulting in attacks because a human discloses information, but also in attacks because the human just enters the same information on the wrong device or accepts a received message he should not. They successfully applied their model to analyse some authentication protocols. Although their approach is similar in spirit to mine, there are some fundamental differences along with some affinities. The two main differences are the following ones. First, they only consider scenarios in which the Dolev-Yao attacker actively attacks the protocol, whereas my approach works also when the attacker is not present thanks to the matching mutations. Second, similar to what I do, they also consider an add rule that allows humans to send "controlled" messages that are in their current knowledge, but my mutations allow us to capture a different, and to some extent wider, set of human deviations from the original ceremony.

Similar to [16], Curzon et al. [50] propose a formal human model that includes a specific attacker able to exploit the errors against the human user. The errors considered are those caused by the humans' interpretation of the system and by the design of the interfaces, but not those entailed by human choices or mistakes as I do. Moreover, they do not consider communication channels.

Johansen and Jøsang [84] define probabilistic processes to model the actions of a human agent, separating the model of the human and that of the user interface. They introduce a "compilation" operation in order to capture the interaction of the human agent and the user interface. Their probabilistic model for the human agent is an extension of the *persona model* [145]. Their approach provides only a preliminary formalisation without a security analysis.

Beckert and Beuster [18] provide a formal semantics for GOMS models augmented with formal models of the application and the user's assumptions about the application, but they do not consider human mistakes in detail.

Pavlovic and Meadows [125] employ *actor-networks* as a formal model of computation and communication in networks of computers, humans and their devices, but they too do not consider human mistakes in detail.

Radke and Boyd [130] introduce the notion of *human-followable security* wherein a human user can understand the process and logic behind authentication protocols. They focus on showing how to transform existing authentication protocols into protocols with human-followable security.

While my approach is quite radically different from the research in [50, 18, 84, 125, 130], I believe that there might be interesting synergies between my mutations and the way in which they model the assumptions and perceptions of the human users, which I plan to investigate in future work.

5.10 Conclusions

My approach allows security analysts to consider human "shades of gray" in the analysis of security ceremonies. I have already mentioned a number of directions for future work. I believe that the most interesting ones are: extend the current mutations by weakening some of the constraints (e.g., on the types and formats of the messages, say to consider other controlled notions of "sendable" message or the case in which the right message is composed in a wrong format); consider other abilities of the attacker (e.g., as in [9]); extend X-Men's library of behavioural patterns with other mutations; formalise combinations of mutations and prove compositionality results; improve the efficiency of my approach by reducing the number of generated mutated models (e.g., by identifying isomorphic models) and by automatically checking whether attacks are real or not; link my formal analysis to mutation testing by generating test cases out of the attack traces; and, finally, consider other, even more complex, case studies.

Chapter 6

A Socio-Technical Approach for Free Travel: How to Exploit Human Ticket Inspection in Coach Services

In this chapter, I introduce a socio-technical study that I carried out considering coach services operating in different countries. This work has not been published yet, but a paper is in preparation.

6.1 Motivation

People come and go from and to airports thanks to several transportation services: underground, trains, taxis, coaches and buses. These public transportation services require the customer to buy a ticket in order to use the service (or to pay a bill in case of a taxi, which I will not consider here). Many transportation companies operate using a similar ticketing system. They accept tickets either in their physical version

or in their electronic version, to allow customers to have continuity with the most common "paper versions" but also to take advantage of technology for more convenient and easy-to-use alternatives. Introducing new types of ticket (e.g., e-tickets, smart cards, etc.) goes hand in hand with designing the corresponding process to check the validity of a ticket. Because of the multitude of physical and electronic versions, as well as the variety of transportation services all over the world, a ticket inspection ceremony should be tailored to the type of ticket used for the particular service.

Forging represents a serious problem for transportation services. For instance, reporters of the BBC [17] were able to buy forged train tickets on the dark web and use these tickets, even though the "technological part" of the tickets (i.e., the magnetic strip) was not valid, to convince the train inspector to allow them to travel. So, driven by this significant issue, I have used my formal approaches described in Chapter 4 and Chapter 5 to analyse the security of a ticket inspection ceremony used in coach services.

6.2 Contributions

In this chapter, *I investigate a security ceremony that defines the inspection of tickets in coach services to check whether the ceremony is secure against forging attacks (which it is not).* More specifically, I provide four main contributions.

1. Hands-on observations I considered and observed a number of coach services, gathering insights on *(i)* the design of their tickets, *(ii)* how they perform their inspection ceremony, *(iii)* the conditions in which the ticket inspection is carried out.

2. Practical methodology of attack I have devised a practical methodology to exploit the inspection ceremony, showing how an attacker can forge tickets for

travelling for free. The methodology allows one to forge e-tickets for coach services in a very simple way, which can be easily replicated, even by laypersons, as it does not require particular technical competence but at most some familiarity with photo-editing software such as Adobe Photoshop or Gimp.

3. Formalization I have defined the attacker model based on the charting technique as shown in Chapter 4, and I then formalised the security ceremony that represents the inspection procedure carried out by the inspector (e.g., the driver) extending the approach described in Chapter 5 with a new human mutation.

4. Security analysis I have defined a security property that models the correct execution of an inspection ceremony and highlighted possible security failures in a preliminary security analysis. I have carried out a security analysis using Tamarin, discovering a potential threat on the ticket inspection ceremony. This has led me to the identification of a series of empirical principles that can improve the security of the ceremony I considered.

For concreteness, I focused my attention on one of the most convenient and reliable coach services (which I will keep anonymous for ethical reasons) that counts millions of passenger journeys every year. However, my analysis is general and independent of that service and could be applied similarly to other services (other coach services but also ticket inspections for cinemas, theatres, concerts, events) that likely suffer from similar problems.

6.3 Outline

The coach service ecosystem is defined in Section 6.4, where I explain the different types of tickets, provide a description of the ticket inspection ceremony I have identified,

and then formalise the ticket inspection ceremony using, and extending, the formal methodology developed in Chapter 5. The formalisation and the new human mutation are in Section 6.5. In Section 6.6, I explain how to carry out the two phases of the attack, describing how to forge an e-ticket and showing, to prove my insights, the results of the formal verification on the ticketing inspection ceremony. Taking stock of what I have observed, I discuss, in Section 6.7, some key factors of the design of the ticket inspection ceremony I analysed, and then I identify two principles that can improve the design of new inspection ceremonies. In Section 6.9, I discuss related work. In Section 6.10, I draw conclusions and discuss future work.

6.4 The Coach Service Ecosystem

In this section, I define the ecosystem of coach services. I begin by explaining the different types of tickets considered, then present the ticket inspection ceremony that I have derived empirically by carrying out a number of journeys, and finally discuss the security property that pertains to the authenticity of the tickets.

6.4.1 The Tickets

The majority of coach services offer essentially two types of tickets: *paper tickets* and *electronic tickets*, or *e-tickets* for short — I do not consider smart-card tickets because they would require a different kind of attack and, anyway, smart-card solutions are not widely adopted.

Customers can buy fixed-price paper tickets (e.g., similar to the tickets in Figure 6.1) physically in shops at the airports, in coach stations or directly from the coach driver on the day of travel.

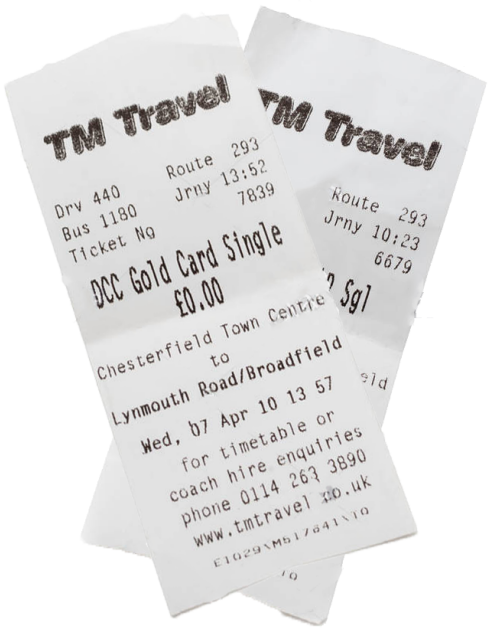


Figure 6.1 Example of paper tickets of a coach service

Alternatively, they can buy e-tickets online on the website of the coach service. The procedure to buy an e-ticket requires the customer to choose the journey specifying the date of travel (or dates in case of a return ticket, unless it is an "open" return ticket, which has a flexible validity), insert the customer's details and pay for the ticket. Once the payment is concluded, the customer receives a confirmation email for the payment, which also acts as the e-ticket for the journey.

Different coach services offer e-tickets that share many similarities in terms of information present on the ticket itself. I considered these similarities to define a standalone e-ticket as shown in Figure 6.2, which is designed based on the information the different tickets of the different coach services have in common. This is fine as it is realistic enough to showcase the main features of my approach.

A standalone e-ticket has the following fields: *customer name*, *ticket number*, *departure date*, *return date* (if it is a return ticket), *departure time*, *arrival time* (or an estimate arrival range time), *from* (which specifies where the journey starts), *to* (which specifies where the journey ends) and the *price* of the ticket. In case the e-ticket is

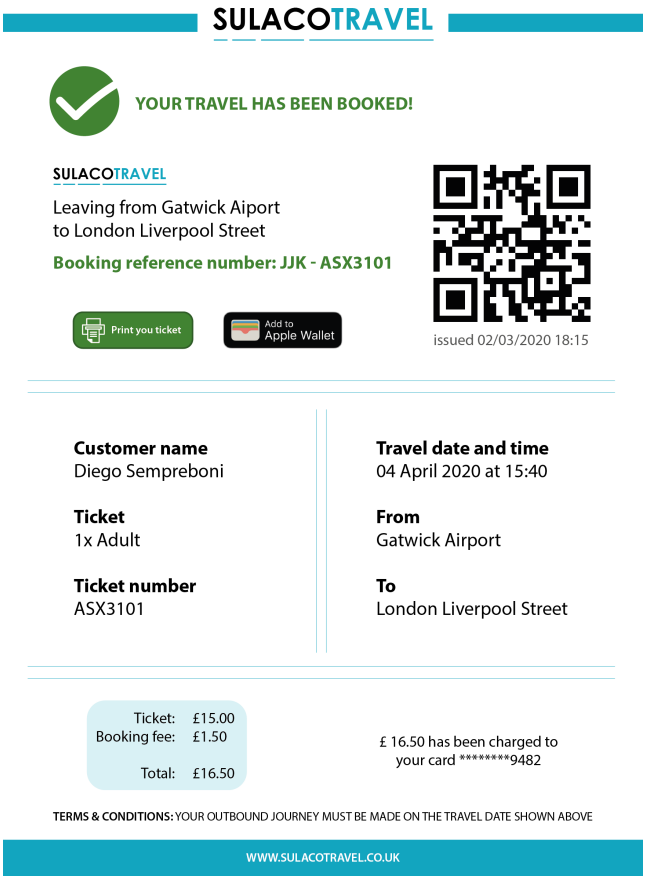


Figure 6.2 The standalone e-ticket based on the e-tickets similarities of a number of coach services analysed

a return ticket, there are additionally fields for the return departure time and return arrival time, from/to.

A number of coach services (as well as other services that offer e-tickets) enrich the e-ticket with some visual field like a barcode or a QR code: these aim to simplify and speed up the ticket inspection process. To that end, the driver is equipped with a device capable of recognising the information within the codes so that such information does not have to be controlled manually.

The e-ticket may also contain other useful information and the terms of condition, which I have omitted here as they are not relevant for the description nor for the attack.

	paper ticket	**e-ticket**	**forging**
Date	×	×	Must be modified
Departure time	×	×	
Arrival time	×	×	
From	×	×	May have to be modified
To	×	×	
Service	×	×	
Price	×	×	
Visual field/s		×	Cannot be modified (require a much more complicated attack)
Ticket number	×	×	

Table 6.1 Fields of a generic paper ticket and of a standalone e-ticket

Table 6.1 show the fields of a generic paper ticket compared with the fields of an e-ticket (the last column will be discussed later).

6.4.2 The Ticket Inspection Ceremony

There is no available generic and public specification for the inspection ceremony of coach services. Thus, I derived the steps that form a plausible ceremony empirically. I analysed the type of tickets available and their fields, highlighting similarities and differences. In parallel, I conducted real observations travelling for a total of 40 times using real tickets and a number of different coach services in order to understand the various ticket inspection ceremonies and whether there are substantial differences between them. Once I collected enough experience and data on the different ceremonies, I synthesized them into a single plausible inspection ceremony, which comprises two sub-ceremonies: one for paper tickets and one for e-tickets.

Observations data

Before I explain the two sub-ceremonies, let me discuss the data collection process.

I have travelled for a total of 40 times with a number of different coach services between April 2017 and December 2019 as in Table 6.2 (dates are reported in *mm/dd/yyyy* format). The journeys were carried out at different hours of the day and different months and years in order to guarantee heterogeneity of the conditions of the traffic, the drivers and the coach services in general. This allowed me to get a broad view of possible variations of the ticket inspection ceremony carried out in order to include these variations, if relevant, in my analysis.

The Paper Ticket Sub-Ceremony

The paper ticket sub-ceremony is shown in Figure 6.3. The driver receives the paper ticket from the customer and then checks the ticket by checking information such as *from/to*, the *departure time*, and eventually the *coach number*. If this information is correct, then the driver admits the customer on the coach. Note that there is no check on the genuineness of the ticket.

The E-Ticket Sub-Ceremony

The e-ticket sub-ceremony, which is shown in Figure 6.4, keeps the same spirit of the paper ticket ceremony, but delegating some of the checks to technology (or so one would expect). Here too, upon request of the driver, the customer shows the e-ticket that comes as the confirmation email of the payment for the journey. The driver performs a preliminary analysis of the e-ticket to understand the type of the e-ticket checking also the presence of possible visual fields. Based on the presence of visual fields, the driver can make two different choices in order to validate a ticket:

Departure	*Return*	*Type*
04/28/2017	-	Single
06/01/2017	-	Single
08/07/2017	09/15/2017	Return
10/15/2017	-	Single
12/18/2017	01/14/2018	Return
05/02/2018		Single
05/10/2018		Single
05/15/2018		Single
05/27/2018		Single
05/27/2018		Single
06/20/2018		Single
07/04/2018		Single
08/16/2018		Single
09/03/2018		Single
09/20/2018	09/21/2018	Return
10/17/2018		Single
10/21/2018		Single
12/15/2018	12/27/2018	Return
02/16/2019	02/21/2019	Return
02/28/2019	03/03/2019	Return
05/16/2019		Single
05/22/2019		Single
07/25/2019	-	Single
12/19/2019	12/31/2019	Return

Table 6.2 The log of my journey with coach services. The log is reduced at minimal considering possible ethical issues.

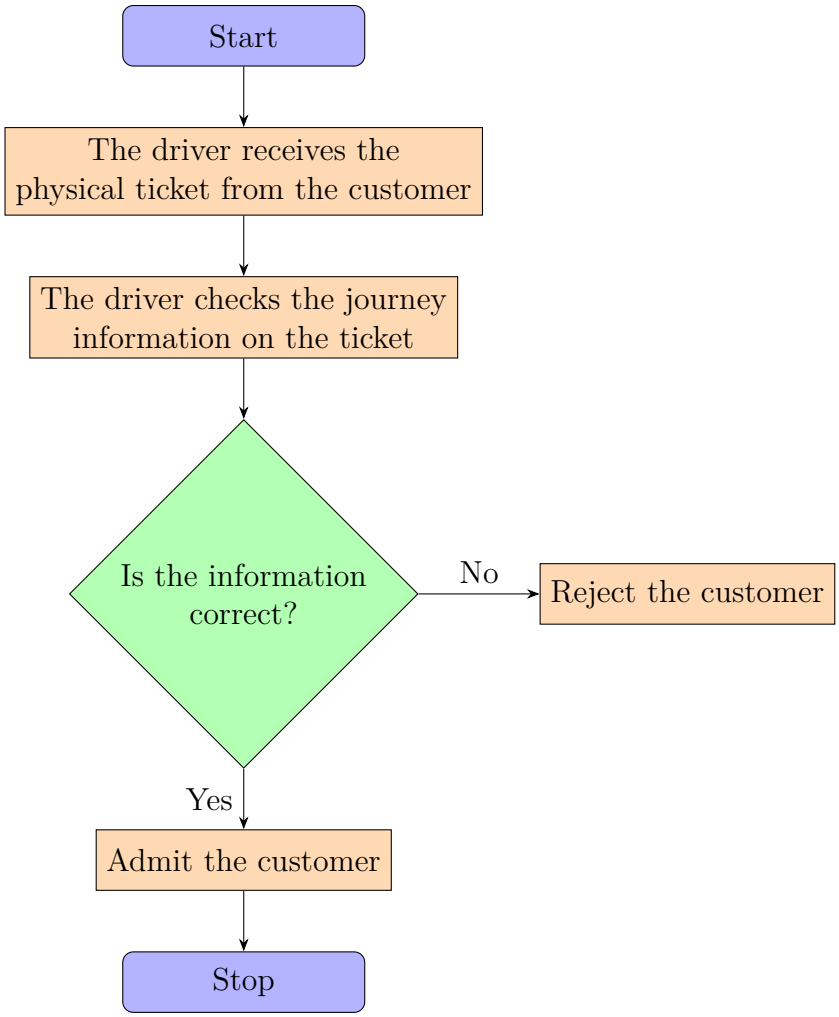


Figure 6.3 The paper ticket inspection sub-ceremony

- check if unique details printed on the ticket (e.g., ticket number) are present on a list of allowed tickets for that journey;
- use the visual fields reading them with an electronic reader.

If the driver decides to check the unique details in the ticket manually, he performs visual inspection, looking for a match of any of the unique details (e.g., ticket number, journey references, service numbers, travel route references, etc.) with a list in his possession (e.g., electronic list using a pad, a printed list). Once he finds a match, the driver validates the ticket and admits the customer. In one of my observations,

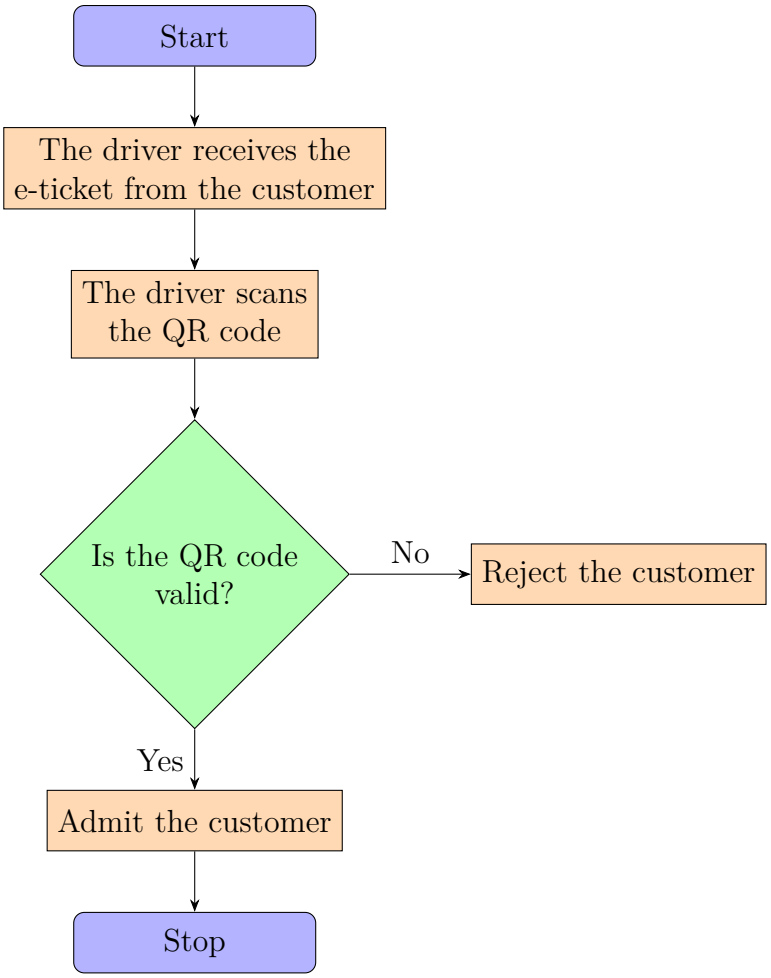


Figure 6.4 The e-ticket inspection sub-ceremony

the driver first identified the ticket number on the e-ticket, and then checked whether the ticket number matched a list of allowed ticket numbers for that journey on an electronic pad (that synchronizes regularly with the main system); once found the match, the driver validated the ticket number to make it unavailable for the future and admitted the customer.

If the driver decides to use the visual fields, he needs to scan them, or a selection of them (e.g., QR code, barcode, etc.), with a device: once the device confirms that the ticket is valid, the driver admits the customer.

6.4.3 Preliminary Security Analysis

A preliminary security analysis of the tickets and of the ticketing inspection ceremony is beneficial to understand possible vulnerabilities. In particular, I looked at flaws that could allow an attacker to travel for free using forged tickets bypassing the ceremony's security measures.

Let's consider the paper tickets, such as the ticket shown in Figure 6.1. Typically, coach services can rely on the difficulty of physically forging accurate copies of paper tickets. Forging a physical ticket would require specific equipment such as the proper paper, a printing machine, the identical layout the coach service is using on a digital file, etc., which are not easy to obtain. This case is similar, in spirit, to how forgers print counterfeit money. These difficulties are reflected in less controls that the sub-ceremony is putting in place as shown in Figure 6.3, where it is possible to see that no specific visual checks are performed, in particular, no checks of unique fields (e.g., ticket number). The checks are limited to the plausibility of the information only.

E-tickets, on the other hand, represent a different scenario. Coach services rely (or should rely, although I proved that it is not the case) on the difficulty of obtaining unique information displayed on the e-ticket to discourage forging. This unique information can also be used by the coach service companies to generate visual fields that might be printed on the tickets.

However, during my analysis, I encountered some visual fields that, even though they could have potentially stored relevant information for the ceremony, they did not contain any information but were just used as graphical means to corroborate the structure of the e-tickets. These visual fields, in fact, were never checked during all my observations. This is a potential attack vector: a coach service that offers customers to choose between different types of e-tickets such as QR code or barcode with no

information (e.g., in case of specific ticket such as discount tickets) can be exploited to execute a successful forging attack.

In the case of e-tickets, the precision with which the checks are carried out is crucial. The inspector (e.g., the driver) has to perform proper checks on the unique fields displayed on the ticket. However, as I have observed, drivers apply the paper ticket inspection sub-ceremony also for e-tickets. From my analysis, the paper ticket sub-ceremony is not designed to be secure against forged tickets if this sub-ceremony is applied in case of e-tickets. As explained above, the paper ticket sub-ceremony does not consider checks on visual fields and relies on the complicated procedure to forge paper tickets. However, the same "security" that is given by the difficult procedure for forging paper tickets cannot be applied in case of e-tickets. I explain how easy it is to forge a coach e-ticket in Section 6.6.2. In fact, assuming a situation where the driver applies the paper ticket inspection sub-ceremony to an e-ticket accurately forged, the attacker sees his forged e-ticket accepted by the driver and is thus able to use the service for free.

6.5 Formalization of the Ticket Inspection Ceremony

To prove my insights, I adopted and extended the formal specification of ceremonies that I gave in Section 5.6 introducing a new mutation that captures a new behavioural pattern.

Let's consider the ceremony described above and shown in Figure 6.5 as a running example. The ceremony has three roles: the *Customer*, the *Driver* and the online website *WebServer* of the coach service.

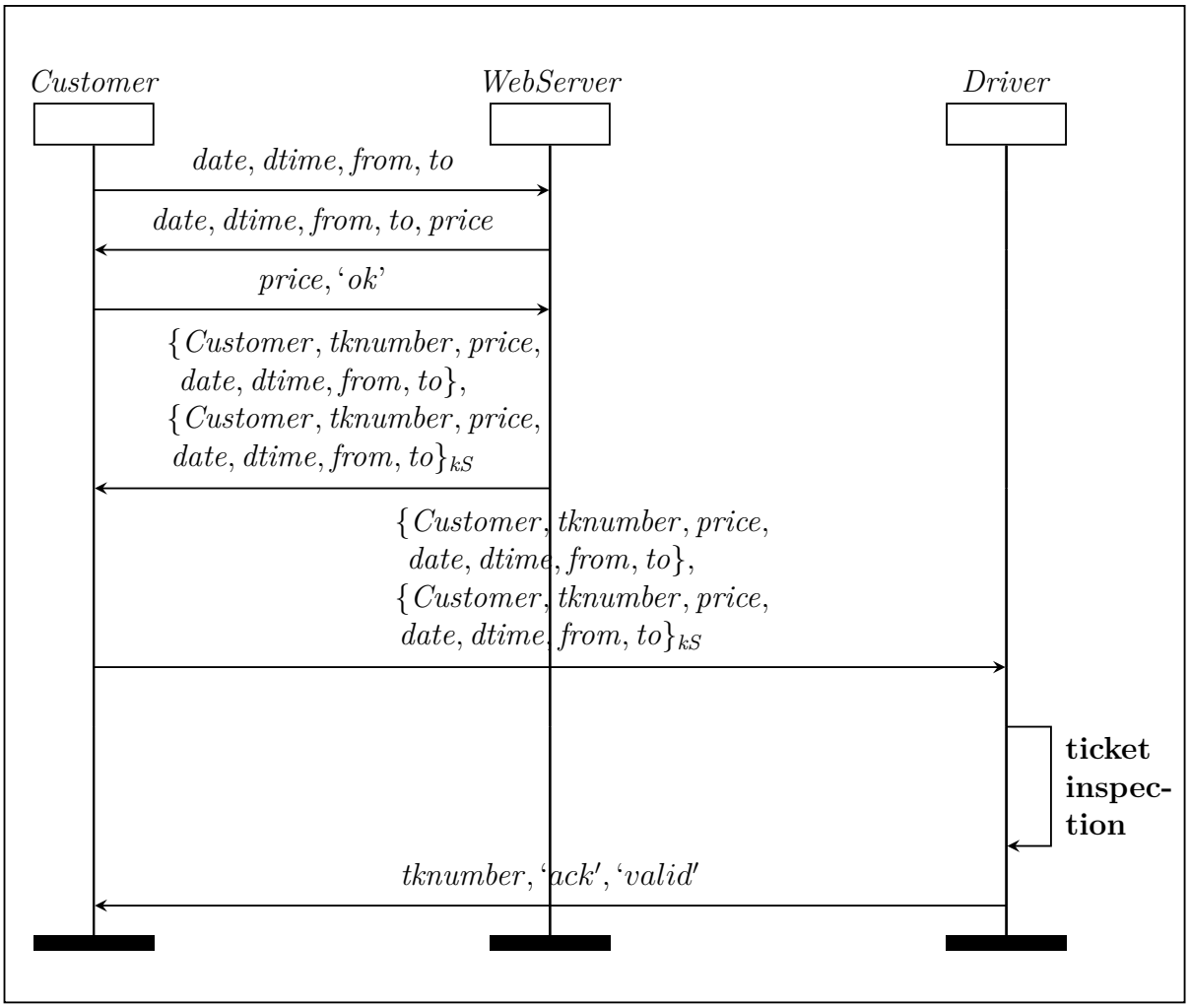


Figure 6.5 The Ticket Inspection Ceremony

1. The *Customer* interrogates the online website *WebServer* in order to get a list of coaches operating on a specific date *date* at a specific time *dtime*, departing from a specific location *from* and arriving to a specific destination *to*.

2. The *WebServer* returns the available solution, indicating the price *price*.

3. The *Customer* clicks on the selected option, paying the price of the ticket.

4. The *WebServer* sends to the *Customer* the e-ticket with the related information, encrypting them using a shared key *kS* between the *WebServer* and the *Driver*. Encrypting the fields represents the characteristic of the e-ticket of being recog-

$$[] \xrightarrow{Start(Customer,\langle WebServer,Driver\rangle)} [\mathsf{AgSt}(Customer, 1, \langle WebServer, Driver\rangle)] \qquad (H_0)$$

$$[\mathsf{AgSt}(Customer, 1, \langle WebServer, Driver\rangle)] \xrightarrow{Snd(Customer,\mathrm{sec},WebServer,\langle date,dtime,from,to\rangle)} [\mathsf{AgSt}(Customer, 2, \langle WebServer, Driver, date, dtime, from, to\rangle), \mathsf{Out}_{sec}(Customer, WebServer, \langle date, dtime, from, to\rangle)] \qquad (H_1)$$

$$[\mathsf{AgSt}(Customer, 2, \langle WebServer, Driver, date, dtime, from, to\rangle), \mathsf{In}_{sec}(WebServer, Customer, \langle date, dtime, from, to, price\rangle)] \xrightarrow{Rcv(Customer,\mathrm{sec},WebServer,\langle date,dtime,from,to,price\rangle)} [\mathsf{AgSt}(Customer, 3, \langle WebServer, Driver, date, dtime, from, to, price\rangle)] \qquad (H_2)$$

$$[\mathsf{AgSt}(Customer, 3, \langle WebServer, Driver, date, dtime, from, to, price\rangle)] \xrightarrow{Snd(Customer,\mathrm{sec},WebServer,\langle price,\text{`}ok\text{'}\rangle)} \mathsf{AgSt}(Customer, 4, \langle WebServer, Driver, date, dtime, from, to, price\rangle), \mathsf{Out}_{sec}(Customer, WebServer, \langle price, \text{`}ok\text{'}\rangle)] \qquad (H_3)$$

Figure 6.6 The rules for the *Customer* in the ticket inspection ceremony

nised valid from the *Driver*. This could be seen as applying a watermarking on the e-ticket.

5. The *Customer*, at the time of travel, shows the e-ticket to the *Driver*.

6. The *Driver* performs a **ticket inspection** and, if successful, admits the *Customer* to the coach.

I use *Agent rules* to specify agents' state transitions and communications. The rules for the *Customer* in the ticket inspection ceremony are shown in Figure 6.6, the rules for the *WebServer* are shown in Figure 6.7 and the rules for the *Driver* are shown in Figure 6.8.

$$[\mathsf{AgSt}(Customer, 4, \langle WebServer, Driver, date, dtime, from, to, price\rangle),$$
$$\mathsf{In}_{sec}(WebServer, Customer, \langle\langle Customer, tknumber, price, date, dtime, from, to\rangle,$$
$$senc(\langle Customer, tknumber, price, date, dtime, from, to\rangle, kS)\rangle)]$$
$$\xrightarrow{Rcv(Customer,\mathrm{sec},WebServer,\langle\langle Customer,tknumber,price,date,dtime,from,to\rangle,\ senc(\langle Customer,tknumber,price,date,dtime,from,to\rangle,kS)\rangle)}$$
$$[\mathsf{AgSt}(Customer, 5,$$
$$\langle WebServer, Driver, date, dtime, from, to, price, tknumber\rangle)] \qquad (H_4)$$

$$[\mathsf{AgSt}(Customer, 5, \langle WebServer, Driver, date, dtime, from, to, price, tknumber\rangle)]$$
$$\xrightarrow{Snd(Customer,\mathrm{sec},Driver,\langle\langle Customer,tknumber,date,dtime,from,to\rangle,\ senc(\langle Customer,tknumber,date,dtime,from,to\rangle,kS)\rangle)}$$
$$[\mathsf{AgSt}(Customer, 6,$$
$$\langle WebServer, Driver, date, dtime, from, to, price, tknumber\rangle),$$
$$\mathsf{Out}_{sec}(Customer, Driver, \langle\langle Customer, tknumber, date, dtime, from, to\rangle,$$
$$senc(\langle Customer, tknumber, date, dtime, from, to\rangle, kS)\rangle)] \qquad (H_5)$$

$$[\mathsf{AgSt}(Customer, 6, \langle WebServer, Driver, date, dtime, from, to, price, tknumber\rangle),$$
$$\mathsf{In}_{sec}(Driver, Customer, \langle Customer, tknumber, date, \text{`}ack\text{'}, \text{`}valid\text{'}\rangle)]$$
$$\xrightarrow{Rcv(Customer,\mathrm{sec},Driver,\langle tknumber,date,\text{`}ack\text{'},\text{`}valid\text{'}\rangle),End(Customer,\text{`}ack\text{'},\text{`}valid\text{'},tknumber,date)} [] \qquad (H_6)$$

Figure 6.6 The rules for the *Customer* in the ticket inspection ceremony

In the rule (D_2) in Figure 6.8, Tamarin actions are used to model the inspection the *Driver* carries out during the ticket inspection ceremony. This check represents a verification of the genuineness of the data encrypted (e.g., using the function *Eq* to check that the fields *Customer*, *tknumber*, *price*, *date*, *dtime*, *from*, *to*, match to their respective encrypted fields of the encrypted e-ticket) during the phase of releasing the ticket by the *WebServer*. Practically, it could consist in checking a watermark on a ticket, or leaving the check to a device that is connected with the *WebServer* itself. To avoid misconceptions, signed fields of the tickets are renamed by affixing $_{kS}$ at the end of the fields.

$$[] \xrightarrow{Start(WebServer,\langle kS,Driver\rangle)} [\mathsf{AgSt}(WebServer, 1, \langle kS, Driver\rangle)] \qquad (WS_0)$$

$$[\mathsf{AgSt}(WebServer, 1, \langle kS, Driver\rangle),$$
$$\mathsf{In}_{sec}(Customer, WebServer, \langle date, dtime, from, to\rangle)]$$
$$\xrightarrow{Rcv(WebServer,\mathrm{sec},Customer,\langle date,dtime,from,to\rangle)}$$
$$[\mathsf{AgSt}(WebServer, 2, \langle kS, Driver, date, dtime, from, to\rangle] \qquad (WS_1)$$

$$[\mathsf{AgSt}(WebServer, 2, \langle kS, Driver, date, dtime, from, to\rangle]$$
$$\xrightarrow{Snd(WebServer,\mathrm{sec},Customer,\langle date,dtime,from,to,price\rangle)}$$
$$[\mathsf{AgSt}(WebServer, 3, \langle kS, Driver, date, dtime, from, to, price\rangle),$$
$$\mathsf{Out}_{sec}(WebServer, Customer, \langle date, dtime, from, to, price\rangle)] \qquad (WS_2)$$

$$[\mathsf{AgSt}(WebServer, 3, \langle kS, Driver, date, dtime, from, to, price\rangle,$$
$$\mathsf{In}_{sec}(Customer, WebServer, \langle price, \text{`}ok\text{'}\rangle)] \xrightarrow{Rcv(WebServer,\mathrm{sec},Customer,\langle price,\text{`}ok\text{'}\rangle)}$$
$$[\mathsf{AgSt}(WebServer, 4, \langle kS, Driver, date, dtime, from, to, price\rangle] \qquad (WS_3)$$

$$[\mathsf{AgSt}(WebServer, 4, \langle kS, Driver, date, dtime, from, to, price\rangle)]$$
$$\xrightarrow{\substack{Snd(WebServer,\mathrm{sec},Customer,\langle\langle Customer,tknumber,price,date,dtime,from,to\rangle,\\ senc(\langle Customer,tknumber,price,date,dtime,from,to\rangle,kS)),\\ ValidTicket(WebServer,Customer,\text{`}tn\text{'},tknumber,date)}}$$
$$[\mathsf{AgSt}(WebServer, 5, \langle kS, Driver, date, dtime, from, to, price, tknumber\rangle,$$
$$\mathsf{Out}_{sec}(WebServer, Customer, \langle\langle Customer, tknumber, price, date, dtime,$$
$$from, to\rangle, senc(\langle Customer, tknumber, price, date, dtime, from, to\rangle, kS)\rangle)] \qquad (WS_4)$$

Figure 6.7 The rules for the *WebServer* in the ticket inspection ceremony

As I advocated above, one can consider a scenario where a *Customer* can actually forge e-tickets in a way that, if some of the forged fields are checked, then they won't be accepted. The part in which these checks are being carried out in the formalisation is represented by the Tamarin actions. The possibility for a driver to forget to check these fields or check them in a wrong way is already captured bys Definition 1 in Chapter 5.

$$[] \xrightarrow{Start(Driver,\langle kS\rangle)} [\mathsf{AgSt}(Driver, 1, \langle kS\rangle)] \qquad (D_0)$$

$$\begin{array}{l}[\mathsf{AgSt}(Driver, 1, \langle kS\rangle),\\ \mathsf{In}_{sec}(Customer, Driver, \langle\langle Customer, tknumber, price, date, dtime, from, to\rangle,\\ senc(\langle Customer_{kS}, tknumber_{kS}, price_{kS}, date_{kS}, dtime_{kS}, from_{kS}, to_{kS}\rangle, kS)\rangle)]\\ \xrightarrow{\substack{Rcv(Driver,\mathrm{sec},Customer,\langle\langle Customer,tknumber,price,date,dtime,from,to\rangle,\\ senc(\langle Customer_{kS},tknumber_{kS},price_{kS},date_{kS},dtime_{kS},from_{kS},to_{kS}\rangle,kS)}}\\ \qquad [\mathsf{AgSt}(Driver, 2, \langle kS, Customer, tknumber, date, dtime, from, to\rangle)]\end{array} \qquad (D_1)$$

$$\begin{array}{l}[\mathsf{AgSt}(Driver, 2, \langle kS, Customer, tknumber, date, dtime, from, to\rangle)]\\ \xrightarrow{\substack{Snd(Driver,\mathrm{sec},Customer,\langle tknumber,date,\text{`}ack\text{'},\text{`}valid\text{'}\rangle)\\ Eq(tknumber,tknumber_{kS}),Eq(Customer,Customer_{kS}),Eq(price,price_{kS}),Eq(date,date_{kS}),\\ Eq(dtime,dtime_{kS}),Eq(from,from_{kS}),Eq(to,to_{kS})}}\\ \qquad [\mathsf{AgSt}(Driver, 3, \langle kS, Customer, tknumber, date, dtime, from, to\rangle),\\ \qquad \mathsf{Out}_{sec}(Driver, Customer, \langle Customer, tknumber, date, \text{`}ack\text{'}, \text{`}valid\text{'}\rangle)]\end{array} \qquad (D_2)$$

Figure 6.8 The rules for the *Driver* in the ticket inspection ceremony

In this mutation, in which one or more Tamarin actions are removed, the mutation is neither matched nor propagated. To clarify this, let us consider the possible cases:

- since we defined a strong correlation between some of the events in Tamarin and some facts[1], the human mutation is not allowed to remove the Tamarin actions that correspond to these facts because otherwise the resulting rule won't be well-formed (moreover, the removal or modification of $\mathsf{Out}_l(A, P, m)$ and $\mathsf{In}_l(P, A, m)$ facts are already covered by the other mutations);

[1]Recall from Section 5.6.3 that, for every event e in the script of a role A, we get a transition rule $prem \xrightarrow{a} conc$ as follows: the label of the rule contains the event, i.e., $e \in a$; *prem* contains an agent state fact $\mathsf{AgSt}(A, step, kn)$, and *conc* contains the subsequent agent state fact $\mathsf{AgSt}(A, step, kn')$, where *step* refers to the role step the agent is in and *kn* is the agent's knowledge at that step. If $e \in a$ is: $Snd(A, l, P, m)$ then *conc* additionally contains an outgoing message fact $\mathsf{Out}_l(A, P, m)$; $Rcv(A, l, P, m)$ then *prem* contains an incoming message fact $\mathsf{In}_l(P, A, m)$; $Fresh(A, m)$ then *prem* contains $\mathsf{Fr}(m)$; $Start(A, m)$ then it is translated to a setup rule where *conc* contains the initial agent state $\mathsf{AgSt}(A, 0, m)$.

- the human mutation is also not allowed to remove a Tamarin action that is necessary for a security goal, since otherwise we would obtain a ceremony that is not executable;[2]

- if the human mutation removes one or more Tamarin actions that entail some restriction on the set of traces (e.g., an equality check $Eq(date, date_{kS})$ like in the ticketing example considered here, cf. Figure 6.8), then the actual Tamarin restriction is still valid as it is a generic property of the operator (e.g., of the equality operator Eq) but it simply will not be applied (as $Eq(date, date_{kS})$ is not present anymore) and the X-Men tool will thus analyse more traces, possibly including some traces that contain an attack based on the mutation;

- in the role scripts of the other agents there is nothing that corresponds to the events of another agent (so the other agents will not notice if one or more Tamarin actions of the human are removed).

In the following subsection, I will instantiate the generic Definition 1 to define this new human mutation both formally and practically.

6.5.1 The Tamarin Action Mutation

This mutation captures the fact that a human user may not adhere to one or more high-level behaviours expected by the ceremony (e.g., the case in which the *Driver* does not execute a check on a particular field on a ticket during the ticket inspection ceremony). This is defined by the mutation as the removal of one of more Tamarin actions.

[2]Another possibility would be to leave the mutation dealing with the removal of the Tamarin actions necessary for a security goal. I leave the investigation of this further aspect to future work.

Definition 5. *A* Tamarin action mutation

$$\mu^{H}_{action} : tr \rightarrow tr'$$

is a human mutation of tr's human subtrace $[a_0, \ldots, a_i, \ldots, a_n]^H$ *such that* tr' *includes the new human subtrace* $[a_0, \ldots, a_{i-1}, [\![a_i]\!]^{\mu}, [\![a_{i+1}]\!]^{\mu}, \ldots, [\![a_k]\!]^{\mu}, a_{k+1}, \ldots, a_n]$*, where* $\{a_i, a_{i+1}, \ldots, a_k\} = a$ *are the Tamarin actions in the human transition rule* $prem \xrightarrow{a} conc$ *that has been mutated by the human into* $prem \xrightarrow{[\![a]\!]^{\mu}} conc$*.*

In the specific case that I am considering in this chapter, the mutation of a Tamarin action in a transition rules $prem \xrightarrow{a} conc$ means either removing it from a or leaving it unchanged, but, in general, it could mean also modifying it by changing some of its parameters.

In Section 5.7, I show the results of the mutations on the rules using abstract "merged" transition rules, in the style of multiset rewriting. This representation highlights only *prem* and *conc* of the rules, which is not suitable for showing the results of this new mutation. So, in order to show the new mutation, I need to consider the full *Agent rules* specified in Figure 6.8. In particular, let's consider the rule (D_2), which is modified according to the mutation as in Figure 6.9.

This mutation reflects a mistake of the *Driver* who forgets to control that the date printed on the e-ticket shown by the *Customer* is, in fact, the original date of the journey for which it was bought. The removal of this Tamarin action will entail that all the possible traces where this check is not valid are considered during the analysis of the security goal (whereas these traces would have been excluded in presence of the check, thus preventing the forging attack that I discuss here).

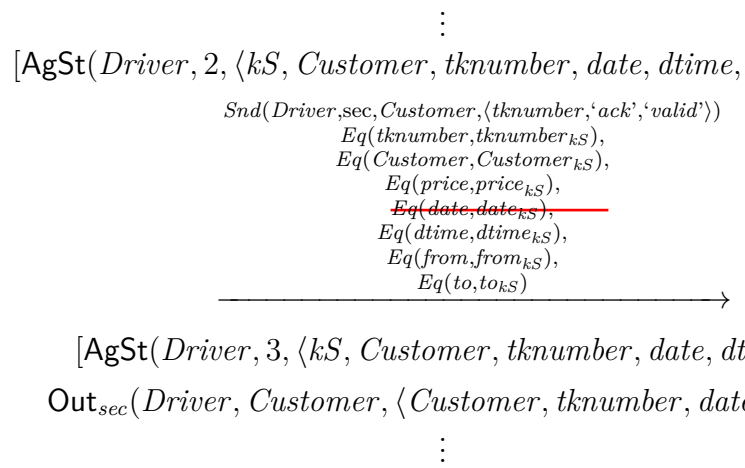


Figure 6.9 Example of the Tamarin action mutation for the ticket inspection ceremony

6.5.2 Security Properties

I consider one security property: *authenticity of the ticket*. Other properties could of course be considered (e.g., ensuring that customers are billed properly or the other security properties for the London underground billing system considered in Chapter 5), but they will be modelled and analysed similarly.

The goal can be formalised by the lemma in Listing 6.1.

```
lemma auth: all-traces
  "All Customer tknumber date #i.
   End(Customer,'ack','valid',tknumber,date)@i
     ==> Ex Driver #j.
         ValidTicket(Driver,Customer,'tknumber',
         tknumber,date)@j & j<i"
```

Listing 6.1 Lemma for the authenticity of the ticket

A ticket inspection ceremony verifies the lemma `auth` if, considering a `Customer` who completes the ceremony with a specific ticket `tknumber` valid for a specific `date`, recognised as `'valid'` (action `End(Customer,'ack','valid',tknumber,date)` at time `i`), then there is a previous time instant `j` such that the specific ticket `tknumber` was issued by the *WebServer* (action `ValidTicket(Driver,Customer,'tknumber',tknumber ,date)` for the specific `date` and `Customer`. This goal refers to a single ticket inspection ceremony, i.e., a single ceremony session, that comprises also the purchase phase in which that specific ticket has been bought.

6.5.3 Threat Models

As discussed in Section 6.1, forging represents a serious problem for transportation services. Companies have to deal with dishonest passengers who try to exploit different vulnerabilities in order to travel for free. Reasoning on the principals that play a key role in the ceremony is then necessary, allowing for the identification of other scenarios. Using the approach formalised in Chapter 4, I can also prove that the assumptions I did in Section 6.4.3 are legit. In the ceremony, I identify the following principals:

$$
\begin{aligned}
&\mathit{Humans}(\mathit{Coach\text{-}Ticketing}) := \{\mathit{Customer}, \mathit{Driver}\}\\
&\mathit{Technicals}(\mathit{Coach\text{-}Ticketing}) := \{\mathit{WebServer}\}\\
&\mathit{Principals}(\mathit{Coach\text{-}Ticketing}) :=\\
&\qquad \mathit{Humans}(\mathit{Coach\text{-}Ticketing}) \ \cup \ \mathit{Technicals}(\mathit{Coach\text{-}Ticketing})
\end{aligned}
$$

The most general case in which every principal gets all possible labels (cf. Section 4.4.4) sees:

$$
\begin{aligned}
n(\mathit{PLHumans}_{\mathit{Coach\text{-}Ticketing}}(\mathit{Customer})) &= 4\\
n(\mathit{PLHumans}_{\mathit{Coach\text{-}Ticketing}}(\mathit{Driver})) &= 4\\
n(\mathit{PLTechnicals}_{\mathit{Coach\text{-}Ticketing}}(\mathit{WebServer})) &= 4
\end{aligned}
$$

Hence, the full threat model chart has $4^3 = 64$ lines.

I focus on one threat model derived from my chart. The thread model (a) in Table 6.3 considers a human *Customer* who chooses to forge an e-ticket to travel for free. This threat model also sees the human *Driver* mistakenly check the ticket using the paper inspection sub-ceremony, so avoiding any check on unique fields. Alternatively, this scenario could be seen as the human *Driver* deciding to only perform a visual check on the ticket and *not* to scan the QR code. Like the Danish Mobilpendlerkort described in Section 4.6, this vulnerability does not require an attacking third party because the human *Customer* takes advance of an inspector who applies the wrong sub-ceremony.

	Human	*Driver*	*WebServer*
(a)	*choice*	*error*	*normal*

Table 6.3 The threat model considered in the ticketing inspection ceremony.

6.6 The attack in a Nutshell

This section describes an attack that exploits the security weaknesses of the ticket inspection ceremony. I have discovered this attack by following, manually, the approach of X-Men; one of the main reasons for this choice is that, in addition to the implementation aspects, this preliminary analysis should be complemented by field experiments interacting with the coach services to carry out the attack for real, but this will require both ethical approval by all stakeholders and the time to carry out the experiments.

6.6.1 The Two Phases of the Attack

The attack to the inspection ceremony is carried out in two phases as shown in Figure 6.10. The first phase, which proceeds as I explain below in Section 6.6.2, requires

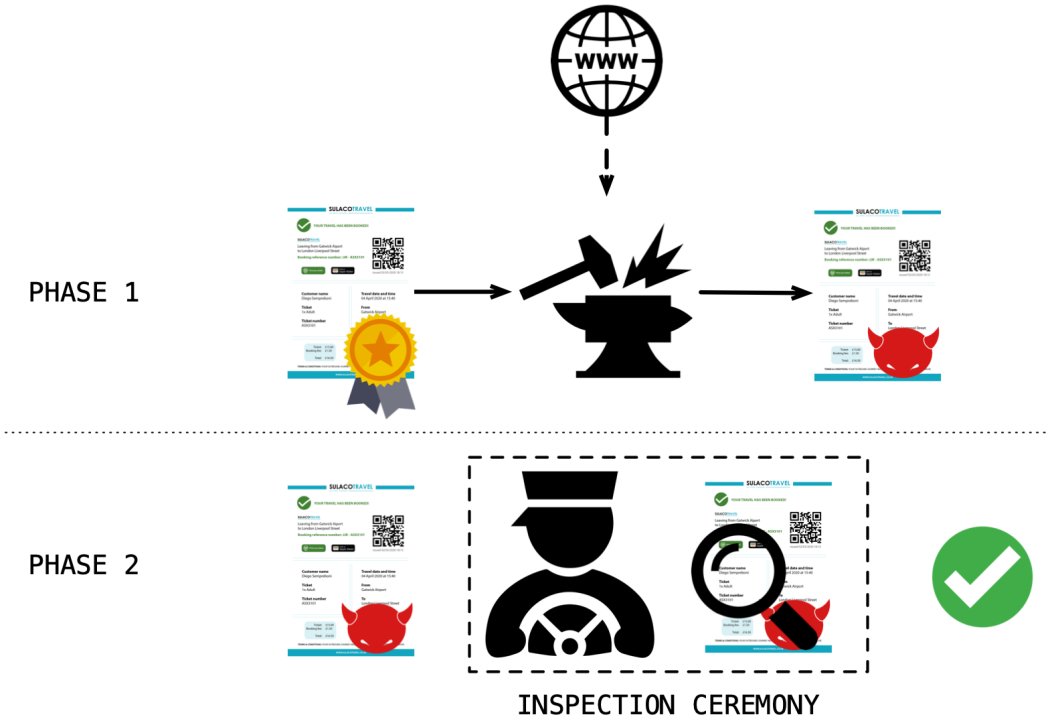


Figure 6.10 The phases of the attack

the creation of a forged e-ticket. I discuss how to forge e-tickets considering the coach services I analysed, but, in principle, the same or a similar approach can be applied for any coach service that uses the same or a similar inspection ceremony. In fact, this approach can potentially be applied to all services that operate using electronic ticketing.

The second phase uses the forged e-ticket as a valid e-ticket for the planned journey, exploiting vulnerabilities on the ticket inspection ceremony. As proof-of-concept, as described in Section 6.6.3, I have used Tamarin to formally and automatically discover the attack on the ceremony.

6.6.2 Forging an E-Ticket

As I highlighted in Section 6.4.3, forging paper tickets is not easy. It requires one to be able to use the same paper type and format, and to print on the ticket the

required information. Still, it can of course be done, as described, for instance, by BBC reporters [17]. Forging e-tickets, on the other hand, is considerably easier and does not require much expertise or technical knowledge. Using a *raster graphics editor* such as Adobe Photoshop (but the same can be done using simpler editors like GIMP) or directly with an *email client*, I was able to forge e-tickets ready to be used in a ticket inspection ceremony. But let's proceed step-by-step.

After having purchased an e-ticket, a confirmation email is sent to the customer's email address. This email also works as ticket, which can be also be saved/exported as PDF file since most of the email clients allow to do it.

There are two ways in which an attacker can modify an existing, old ticket into a forged one for another journey. The most accurate way to forge an e-ticket is by modifying an existing one after having exported, or received, it in PDF format, which provides high-quality graphic details even after a modification. Both Adobe Photoshop or Gimp work.

Alternatively, a less elaborated method to forge e-tickets is for the attacker to modify the confirmation email using the email client and to forward the new email to the attacker's own address.

Regardless of which of the two ways the attacker uses (using PDFs or directly modifying the email by forwarding it), in order to modify an existing, old ticket into a forged one for another journey it is necessary to change fields according to the new journey. Let's consider Table 6.1 again. The last column shows the fields, or combination of fields, that must be modified, can be modified and cannot be modified (without executing a more complex attack such as an attack that creates a new QR code or forges an e-ticket from scratch after obtaining the necessary fields).

The minimal modification that needs to be carried out on an existing e-ticket is to modify the date of travel and/or the departure and arrival time. All the other fields

could in principle be left unchanged, but an attacker may have to modify some of them according to the specific journey that they wish to carry out. The ticket fields that may have to be modified are: from/to, the coach service and the price. For instance, if an attacker wishes to modify an existing ticket from A to B, they will need to modify the date of travel and/or the departure and arrival time, but they will not need to modify the fields from/to, the coach service and the price. If the attacker instead wishes to travel from C to D, then they will have to modify also these other fields accordingly (including the price, which might vary).

There are also some fields that I decided to label as "cannot be modified" as they can only be modified through the execution of a considerably more complicated attack. Ultimately, a skilled attacker could create a forged e-ticket from scratch, but here I am interested in showing that even a completely unskilled attacker can forge an e-ticket by modifying and forwarding the email they received and that a slightly more skilled attacker could use photo-editing software to modify a PDF file. The fields that are labelled as "cannot be modified" are the QR code (if present) and the ticket number and possibly any other field that is generated using some private information such as coach services private keys or unique codes. For instance, in one of my observations, two fields show journey references generated freshly and stored in the system when an online ticket purchase is made. The QR code is also generated freshly using those fields and other ones (the ticket number, a hexadecimal sequence, etc.).

So, without considering an attacker able also to hack the system in order to get this additional information, an e-ticket forged from an old one is an e-ticket that contains some fields that belonged to the old e-ticket but are not valid any more, and some fields that the forger was capable of obtaining because they are public. The vulnerability of the inspection ceremony relies on the driver who does not always properly check the old fields.

This public information can be easily obtained by simulating the purchase of an e-ticket, i.e., by starting a purchase but without completing it. In fact, the online booking system guides the user through a purchase and asks the user to enter the departure location and the destination, the date of travel and the preferred time. The system then finds the more appropriate alternatives showing the departure time, the arrival time, the coach service and the price for each of the solutions suggested. With this information, the attacker is now able to forge an old e-ticket into an e-ticket for that specific journey.

As a concrete example, let us consider the e-ticket in Figure 6.2 for a journey from Gatwick Airport to London Liverpool Street on April 2020.

In this e-ticket, all the fields that have to be considered for possible modification are highlighted with red circles. Now, assume that the attacker wishes to use this ticket to forge one for the same journey but at a different time on a day in May 2020, resulting in the e-ticket shown in Figure 6.11. In this case, the attacker needs to modify the date of travel and the departure and arrival time, retrieving the information through the online booking system, as described above. As explained in Table 6.1, fields such as the price, from, to, could be modified (but not in this case since the price for the same journey has remained the same in the two months that have passed and from/to fields do not need a modification since the journey starts and stops from the same locations), fields such as the ticket number and visual fields like the QR code cannot be modified because they would require a more complicated attack (they require us to hack the booking system in order to generate fresh values for them).

This e-ticket has been modified using Adobe Photoshop after having exported the confirmation email (and so the e-ticket) in PDF. However, as I said, the same modification that I did could have been done directly using an email client. I do not show how to forge an e-ticket by modifying an old one using the email client as the

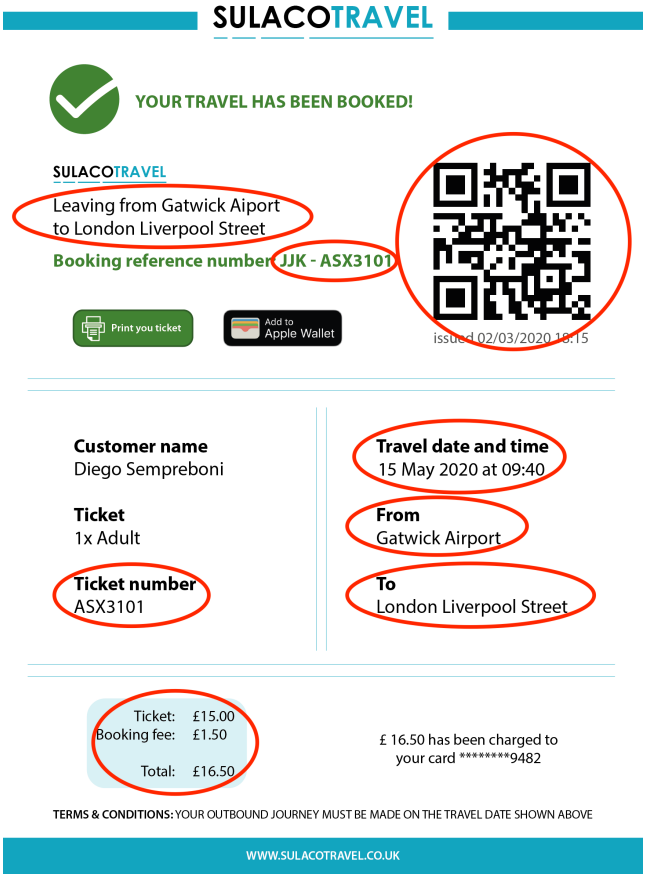


Figure 6.11 The e-ticket after the forging process

approach I have discussed is ultimately independent of how the modification is carried out (raster graphics editor or email client) and the discussion below can be easily adapted to the case in which the email client is used.

At the end, the attacker sends to the attacker's own email address the forged e-ticket as an attachment of a confirmation email copying also the original subject of the email received as subject of the new email. In case of a forged e-ticket made directly modifying it through the email client, the attacker needs to forward the new email to himself.

6.6.3 Formal Verification of the Ticket Inspection Ceremony

I wrote a formal model of the ticket inspection ceremony described in Figure 6.5. The specification of the ceremony, which is given in full in Section A.5, includes:

- a phase in which the customer buys a genuine e-ticket (this phase is needed to obtain a first e-ticket that will be modified later, for other journeys) and
- a ticket inspection performed by a driver of a coach service.

I introduce, following the X-Men framework approach, a replace human mutation for the *Customer* agent which consists in replacing the field date on the genuine e-ticket (i.e., as in the scenario described above where the date has been changed from April to May 2020.). The introduction of the new mutated rule for the *Customer* agent makes him *malicious* since he is trying to perform a step of the ticket inspection ceremony using a ticket with an invalid field. Moreover, I introduce the human mutation on the Tamarin actions for the *Driver*, generating a new mutated rule in which the check of the genuineness of the fields is compromised (e.g., the check of the date). The *Driver*, as also shown in Table 6.3, behaves honestly but makes a mistake. The mutation of the *Customer* rule has a side effect on the *Driver* rule since the latter won't check the changes made by in the *Customer* rule. This is different, however, from the idea of matching mutation defined in Section 5.7.

The *Customer* rule before and after the replace mutation is shown in Figure 6.12.

Note that, even though the `~newdate` is generated as fresh value, as if the *Customer* thinks immediately before showing the e-ticket to the *Driver*, it is possible to adopt the same approach used in Chapter 5, where "types" can be defined modelling a situation in which the new journey was already planned (the new data field can be in the knowledge of the *Customer* with the appropriate type).

```
rule H_3:
  let
    ticket = <Client, tknumber, price
    , date, dtime, from, to>
    encTicket = senc{ticket}kS
    msg = <ticket,encTicket>



  in
  [ State(Client,'3',<S, D, date, dtime
  , from, to, price>)
  , RcvS(S,Client,msg)

  ]
  --[]->
  [ State(Client,'4',<S, D, date, dtime
  , from, to, price, tknumber>)
  , SndS(Client,D,msg)
  ]
```

```
rule H_3_M:
  let
    ticket = <Client, tknumber, price
    , date, dtime, from, to>
    encTicket = senc{ticket}kS
    msg = <ticket,signTicket>
    ticketForged = <Client, tknumber
    , price, ~newdate, dtime, from, to>
    msgForged =  <ticketForged,encTicket>
  in
  [ State(Client,'3',<S, D, date, dtime
  , from, to, price>)
  , RcvS(S,Client,msg)
  , Fr(~newdate)
  ]
  --[]->
  [ State(Client,'4',<S, D, date, dtime
  , from, to, price, tknumber, ~newdate>)
  , SndS(Client,D,msgForged)
  ]
```

Figure 6.12 The original *Customer* rule (left) and the mutated *Customer* rule (right)

The new Tamarin action mutation is applied to the *Driver* human agent in play. The *Driver* rule before and after the replace mutation is shown in Figure 6.13.

Here, as anticipated in Figure 6.9, the Tamarin action $Eq(date, date_{kS})$ is removed. The consequence is that the restriction defined in the ceremony that restricts the traces to be checked by Tamarin only to those that respect the `Equality` check (cf. Section 2.3 and Section A.5), relaxes the constraints, which entails that Tamarin will consider traces that were not considered before.

I fed my specification into Tamarin, which proved the existence of the attack in which the forged e-ticket can be accepted as valid by the *Driver* and the *Customer* is admitted to use the coach service. In fact, Tamarin is able to find the attack to the security goal *authenticity of the ticket* defined in Section 6.5.2 as also shown in Figure 6.14.

As shown in Figure 6.14, the mutated rules `H_3_M`, which is generated by the replace mutation, and `D_1_M`, which is generated by the Tamarin action mutation, contribute to the attack trace.

6.7 Lessons Learned

As I demonstrated, e-tickets for the coach service ecosystem (such as the many cases I observed) can be forged and the detection is not always happening due to factors that influence this ecosystem.

Humans tend to make mistakes, are clumsy and could be subjected to deceptive behaviours. Using the paper ticket inspection sub-ceremony instead of the proper e-ticket inspection sub-ceremony is an error dictated by old habits: even though a new sub-ceremony is available because of the introduction of e-tickets, the driver that is in charge of inspecting the tickets may apply old habits in a different context.

```
rule D_1:
  let
    tck = <encClient,encTKnumber,encPrice
    ,encDate,encDTime,encFrom,encTo>
    encTicket = senc{tck}~kS
    ticket = <Client,tknumber,price,date
    ,dtime,from,to>
    msg = <ticket,encTicket>
    ack = <tknumber,date, 'ack', 'valid'>
  in
  [ State($D,'1',~kS)
  , RcvS(Client,$D,msg)
  ]
  --[Eq(tknumber,encTKnumber)
  , Eq(Client,encClient)
  , Eq(price,encPrice)
  , Eq(date,encDate)
  , Eq(dtime,encDTime)
  , Eq(from,encFrom)
  , Eq(to,encTo)]->
  [ SndS($D,Client,ack)]
```

```
rule D_1_M:
  let
    tck = <encClient,encTKnumber,encPrice
    ,encDate,encDTime,encFrom,encTo>
    encTicket = senc{tck}~kS
    ticket = <Client,tknumber,price,date
    ,dtime,from,to>
    msg = <ticket,encTicket>
    ack = <tknumber,date, 'ack', 'valid'>
  in
  [ State($D,'1',~kS)
  , RcvS(Client,$D,msg)
  ]
  --[Eq(tknumber,encTKnumber)
  , Eq(Client,encClient)
  , Eq(price,encPrice)

  , Eq(dtime,encDTime)
  , Eq(from,encFrom)
  , Eq(to,encTo)]->
  [ SndS($D,Client,ack)]
```

Figure 6.13 The original *Driver* rule (left) and the mutated *Driver* rule (right)

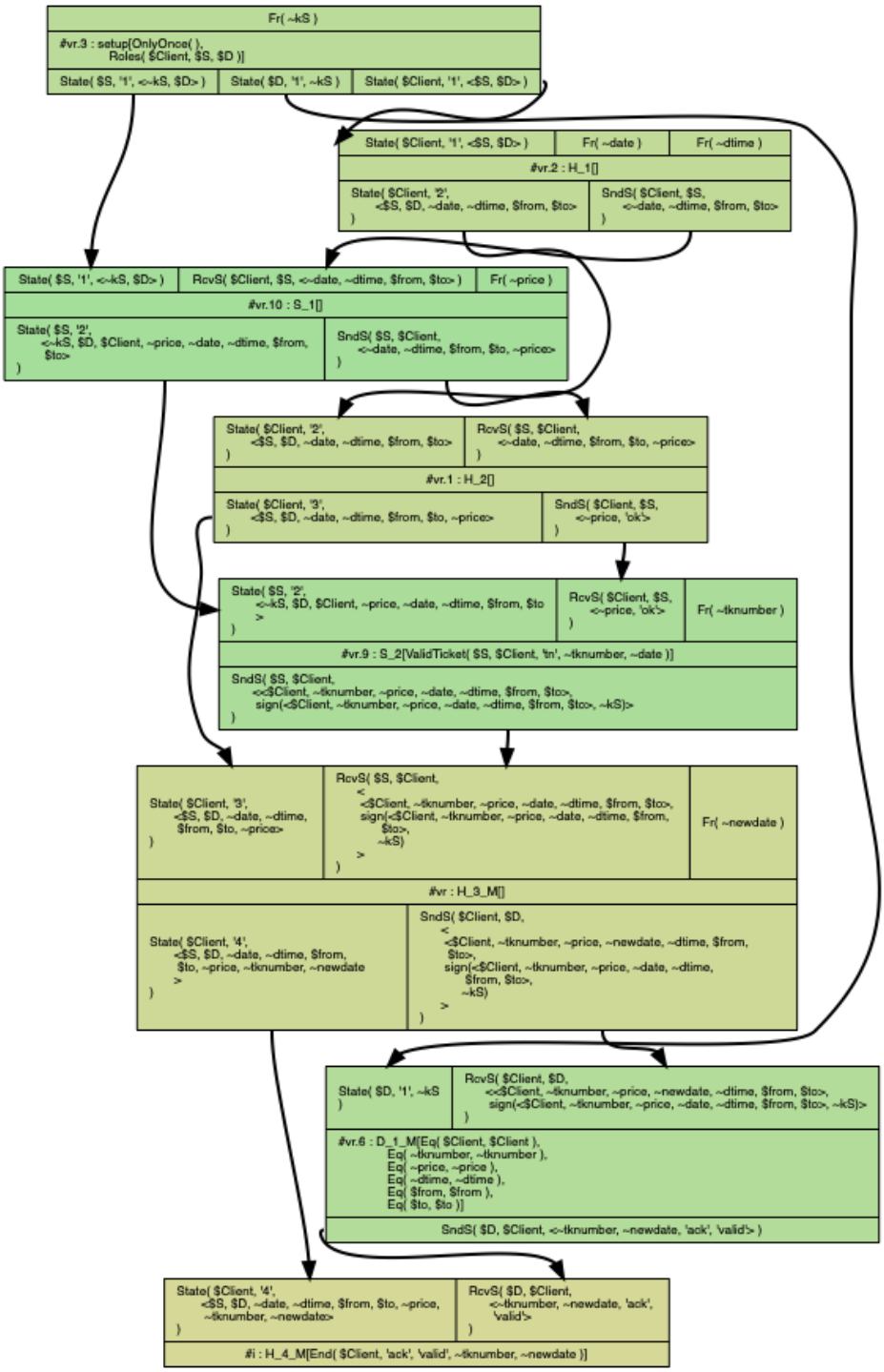


Figure 6.14 The full attack trace found in the Ticket Inspection Ceremony

Moreover, the ticket inspection ceremony has not been simplified over time, removing or merging some checks on tickets. In fact, initially the ticketing inspection ceremony

included only the paper ticket inspection sub-ceremony and when the e-tickets were introduced later on, the ceremony was extended with the e-ticket inspection sub-ceremony, which requires the driver to behave differently. As a consequence, the driver has to know and work through a ceremony that comprises of two different procedures (the one for the paper ticket and the other for the e-ticket and thus might easily confuse the steps of the paper ticket sub-ceremony with the steps of the e-ticket sub-ceremony.

Another factor that plays a role in the weakness of the inspection ceremony is *timing*, which entails two kinds of problems that may cause vulnerabilities.

The first problem is that most of the coach services offer a 24/7 service and drivers who work the night shift might be more vulnerable to be deceived or more inclined to make mistakes. The second problem, which is even more serious, is that coach services operate on strict timetables that must be respected as much as possible; this is especially the case for those services that connect an airport to the city centre. Such coaches are often full (I have seen passengers required to wait for the next bus as the current one was full), which means that the driver must inspect many tickets. The time used to perform the ticket inspection ceremony affects the service and the timetable, so it is in the interest of the driver to spend as little time as possible in checking the tickets. The two sub-ceremonies are different in terms of effort: the paper ticket inspection sub-ceremony does not require any device to set up, and is thus faster than the e-ticket inspection sub-ceremony. This provides an explanation for the fact that a driver under time pressure wrongly applies the paper ticket inspection sub-ceremony to e-tickets. This visual inspection of e-tickets should be secure against forged e-tickets, but it is not because it is designed for another type of ticket (which is much more difficult to forge).

I can thus summarize the lessons learned by identifying the following two principles:

Principle 1. *If the ticket inspection ceremony allows the driver to decide which sub-ceremony to perform, the overall security of the entire ticket inspection ceremony should not depend on the choice of the driver.*

The inspector (i.e., the driver) of a ticket inspection ceremony has to check the validity of tickets and it is up to them how to do it. The final goal is to verify if a ticket is valid or not and provide the service, no matter how. Since the two sub-ceremonies share the same goal but they require different efforts, the inspector may be tempted to apply the quicker/simpler one. Making the two sub-ceremonies unique, in order to avoid that one can confuse the steps of one with the steps of the other, would force the inspector to apply the right sub-ceremony based on the type of the ticket. This principle is similar to one of the design principles proposed by Abadi and Needham in order to reduce the errors in cryptographic protocols [2]. It is also similar to what happens with the composition of security protocols, where part of the messages of one protocol are also used by the other protocol, giving rise to attacks [73, 45, 72, 114, 77, 3].

Principle 2. *In presence of text and visual fields (e.g., barcode or QR code), the information within the visual field has to be redundant.*

By redundant in this case I mean that the visual field contains the information present in the text fields within the ticket. I observed the presence of visual fields (e.g., a barcode) that do not contain any information. Similarly to QR codes, barcodes can, in fact, contain information such as *ticket number* that can be used to validate tickets in the same way as it can be done with QR codes. To respect this principle, the barcode should be made to contain information regarding the ticket.

If the driver decides to carry out the paper ticket inspection sub-ceremony on an e-ticket, the visual field instils a "fake" sense of security on the validity of the e-ticket, the same sense of security a paper ticket gives to the driver when the paper ticket inspection ceremony is carried out.

6.8 Ethics

Socio-technical studies face ethical questions and the research I carried out in this chapter is not far behind. During my observations, I had to take into account the collection of data regarding my journeys and how these data would have been a problem if I used them as proof of my methodology. Reporting my journey log in full would have exposed the drivers to be potentially recognised by the coach service company, especially if I had decided not to anonymise the name of the company.

Another ethical issue would have been to test my insights, especially without the collaboration of the coach services companies. The attack I am describing here requires that the drivers are not aware of checking forged tickets, otherwise this would jeopardise the entire research. However, withouth any agreement with the coach services, this would be considered as fraud, and I could incur into legal problems.

For these reasons, as a first step, I carried out only observations of coach services, carrying out a more in-depth analysis with field experiments once I have established a formal collaboration with the companies. It should be noted that I reached out repeatedly to two coach companies, and even though I managed to speak with one of the CSOs, they stopped the preliminary discussion due to lack of their time. This, however, does not change the importance of my research and the strenght of my methodology showed so far, as I have also proved with a formal verification of my insights. Fields experiments will just confirm what I speculated here, showing that it is possible to exploit these kind of inspection ceremonies.

6.9 Related work

Bella and Coles-Kemp [20] proposed a framework in support of the socio-technical analysis of ceremonies called *the ceremony concertina*. Our analysis considers Layer

III *Human-Computer Interaction*, giving also some insights on Layer IV *Personal* and Layer V *Communal*.

In [66], Garcia et al. carried out a related but different security analysis on ticketing in the UK by considering the MIFARE Classic card, the contactless smart card more commonly known in London with the name of Oyster card. Their analysis is fundamentally different from mine as they reverse-engineered the security mechanisms of the chip discovering two attacks that allowed an intruder to get the secret key of a card in order to clone it. Furthermore, the analysis they carried out is also different from the analysis I carried out in Chapter 5 as I investigated the human component of the Oyster ceremony.

The work that is closest to mine is that of Giustolisi [68], who applied a socio-technical approach to analyse an inspection ceremony for ticketing in Denmark, highlighting how improper generation of mobile transport tickets can lead to forgery attacks. Differently from me, Giustolisi carried out an analysis on the mobile application used in Denmark combining forging techniques on ticketing screens in order to exploit the ceremony. My approach would, of course, not work with a mobile application, but is general enough to be applied to any kind of service that uses the "manual" e-ticket inspection ceremony that I described. Giustolisi identified four principles that helped him to propose solutions to the weaknesses of the ceremony considered, with possible applications to other ceremonies:

Giustolisi's principle 1: The security design of paper tickets should not influence the security design of electronic tickets.

Giustolisi's principle 2: Computer inspection should be prioritised over visual inspection.

Giustolisi's principle 3: The inspection ceremony should enable the verification of ticket key information either electronically or manually.

Giustolisi's principle 4: Security should be preferred over usability in the design of visual inspection of an electronic ticket.

Principle 1 is very valuable, but, as discussed, there are fundamental differences between the spirit with a ticket inspection ceremony is performed in case of paper ticket and in case of e-ticket. The security here is moved from the design of the tickets to the design of the ceremony.

Principle 2 points out the importance of prioritizing a computer inspection and Principle 4 reinforces this view by suggesting that e-tickets should undergo computer inspection instead of visual inspection. These principles are crucial and they should be taught during the training of ticket inspectors (I guess they are, but drivers are under a lot of pressure as I discussed). However, as I showed, old habits have a much stronger bond that makes attacks possible.

Principle 3 suggests a possibility to check tickets also manually, but in my case this has led to the attack, so the ceremony should be spelled out clearly, which is the reason why I encourage a ceremony that relies as little as possible on the human.

My Principle 1 is complementary to the four principles of Giustolisi and can thus contribute to improving the security of a ticket inspection ceremony. My Principle 2 reinforces the position held by Giustolisi's Principle 3 that says that "the inspection ceremony should enable the verification of ticket key information either electronically or manually" by imposing the presence of visual fields that can be controlled electronically (and to allow it, the visual fields must contain the information within the text fields).

6.10 Conclusions

Ticket inspection ceremonies often rely on humans to spot attempts of using forged tickets, and the ticket inspection ceremony of coach services that I considered here, in which the driver checks paper tickets and e-tickets and admits customers on the

coach, is no different. My approach discovered a vulnerability on the ticket inspection ceremony caused by human users. No matter what security experts do to protect a system from attacks, it takes just one user to "ruin everything", and in this case it is the ticket inspector.

As future work, I plan to carry out field experiments to identify the coach services that can be exploited using forged e-tickets, extending the analysis that I carried out and the principles I identified to other security properties. This approach can also be extended to other forms of transportation services and to case studies in other areas where human-based inspection can play a crucial role, such as payment, voting or mobile applications. To that end, I plan to implement the new mutation into my X-Men, which will likely be non-trivial since it will require considering also situations in which the removal of Tamarin actions affects the security goals.

Chapter 7

Conclusions and Future Work

This final chapter provides concluding remarks for the work presented in this thesis (Section 7.1) and points to some interesting directions for future work (Section 7.2).

7.1 Conclusions

This thesis has investigated how to include the *human* in several aspects related to security protocols. Lately, these scenarios have been referred to as *security ceremonies*. The recurring narrative has blamed human users as the weakest link in cybersecurity: no matter what security experts do to protect a system from attack, it takes just one user to "ruin everything". While there is no doubt that a human user can make a system insecure, security experts are often too quick to blame humans rather than the protocols or systems that the engineers have developed. On the other hand, engineers are pushing more and more the checkpoint where unexpected behaviours caused by humans should be managed. This checkpoint nowadays is, often, in the hands of the end-user, the human. The ultimate goal of this thesis was to bring this checkpoint back to where it should be: at the design and analysis phases.

Carrying out user studies through questionnaires was necessary to unveil the psychological and socio-technical aspects that are driving the behavioural choices within security systems (e.g., voting, choosing passwords or not using them at all and how, and accessing procedures). These user studies, strengthened by a semi-formal methodology, successfully helped me in the creation of guidelines to bend the user's perception of cumbersome ceremonies.

However, orienting the design of security ceremonies and gathering insights from end users is not enough. As we know, when considering security ceremonies, in which humans are first-class actors, it is not enough to take this "black&white" view. It is not enough to model human users as "honest processes" or as attackers, because they are neither. Modelling a person's behaviour is not simple and requires formalising the human "shades of grey" that such approaches are not able to express nor reason about. So, in order to increase the "spectrum", I carried out an investigation on what are, and how to define with more granularity, the threat models for security ceremonies. A systematic methodology enriched by a labelling system provided me the appropriate tool to capture the multitude of shades that can be hidden behind a security ceremony.

The charting methodology allows security analysts to approach security ceremony analysis in a systematic way. To enable such an analysis, tool support is called for. The first step in that direction is the definition of a formal language expressive enough to model the wide spectrum of security ceremonies. Besides allowing me to enable reusability of models and results when moving from one case study to another, the formal language I created allowed me to model four human behaviours never considered before in such a detailed way. All these characteristics and the mutations were implemented in a prototype tool *X-Men*, which I have used to analyse the security of old and novel ceremonies.

In conclusion, much effort has been devoted to the technical analysis of the security of ICT systems, and there has also been considerable research on the behaviour of users of technology. However, both the technical analysis and the research on the behaviour of users of technology have, so far, been substantially led by two disciplines (computer science and software engineering, and social sciences and psychology) that have been interacting very rarely in the case of the security of ICT systems, and often don't even speak the same language. I believe that my thesis contributes not only to the concrete socio-technical analysis of security ceremonies but also to bringing these two disciplines closer.

7.2 Future Work

This section makes suggestions on future research directions on the topics covered by this thesis. Let's start by saying that nowadays a variety of real-world ceremonies already exist.

Some of them are "ugly" because designers and developers forgot about end-users. These ceremonies would benefit from being subjected to a refined beautification process based on guidelines such as those proposed in this thesis. However, I am aware that beautifying existing ceremonies may sometimes lead to simplifying them, bringing us to debate about how far we can go into the beautification process without compromising the security. To investigate this aspect, combinations of empirical, analytical and formal approaches need to be developed to help ensure that when the new paradigm is applied to a ceremony, possibly simplifying it, it remains secure. I have already begun working in this direction by extending the work described in [25]. In particular, I have been defining degrees of ceremony attractiveness for humans in a formal, logico-mathematical way, in order to then be able to reason about the interplay of beautification and security, ultimately ensuring that the beauty of a ceremony does not come at the expense of its

security but instead provably reinforces its security. My ambition is to define criteria that formalise when beautification preserves or even reinforces security.

Some of them are insecure and the full threat model chart can help us to discover the potential threats. However, a more fine-grained formalisation would be beneficial for this approach, especially to cope with the size and complexities of the full threat model chart. One approach could be the description of appropriate measures and weights to prioritise the different threat models so as to create an ordered list. The given ceremony could then be verified, in turn, against each item of the list. Another approach could be the definition of methods to handle the full threat model chart in one go, perhaps parameterising the findings upon the threat model.

Some of them are insecure and my tool *X-Men* is able to consider human "shades of grey" in the analysis of security ceremonies. The tool, but also the methodology, can be improved by extending the current mutations by weakening some of the constraints, considering other abilities of the attacker (e.g., as in [9]), extending X-Men's library of behavioural patterns with other mutations, formalising combinations of mutations and proving compositionality results; improving the efficiency of the approach by reducing the number of generated mutated models; automatically checking whether attacks are real or not (meaning that they can be applied concretely on the ceremony's implementation); and, last but not least, linking our formal analysis to mutation testing by generating test cases out of the attack traces.

Of course, all the above future works will need to be complemented with run-time testing to check the ceremonies' security while they are being executed, and they will need to be coupled with more detailed user studies and the identification of behaviour change techniques to investigate the interaction and behaviour of the users with the ceremonies. These studies will improve the formal models, which in turn will improve the ceremonies and thus ultimately the user studies themselves. I believe that this

thesis provides a stepping stone and identifies a roadmap for the full-fledged conjugation of user studies, formal analysis, run-time testing and behaviour change techniques in order to establish the security of ICT systems socio-technically.

Bibliography

[1] Abadi, M. and Fournet, C. (2001). Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, volume 36, pages 104–115. ACM.

[2] Abadi, M. and Needham, R. (1996). Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15.

[3] Almousa, O., Mödersheim, S., Modesti, P., and Viganò, L. (2015a). Typing and compositionality for security protocols: A generalization to the geometric fragment. In *Proceedings of the 20th European Symposium on Research in Computer Security*, volume 9327 of *Lecture Notes in Computer Science*. Springer.

[4] Almousa, O., Mödersheim, S., and Viganò, L. (2015b). Alice and Bob: Reconciling Formal Models and Implementation. In *Programming Languages with Applications to Biology and Security — Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday*, volume 9465 of *Lecture Notes in Computer Science*, pages 66–85. Springer.

[5] Armando, A., Arsac, W., Avanesov, T., Barletta, M., Calvi, A., Cappai, A., Carbone, R., Chevalier, Y., Compagna, L., Cuéllar, J., Erzse, G., Frau, S., Minea, M., Mödersheim, S., von Oheimb, D., Pellegrino, G., Ponta, S. E., Rocchetto, M., Rusinowitch, M., Torabi Dashti, M., Turuani, M., and Viganò, L. (2012). The AVANTSSAR Platform for the Automated Validation of Trust and Security of Service-Oriented Architectures. In *Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 7214 of *Lecture Notes in Computer Science*, page 267–282. Springer.

[6] Armando, A., Basin, D. A., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P. H., Heám, P. C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., and Vigneron, L. (2005). The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Proceedings of the 17th International Conference on Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, page 281–285. Springer.

[7] Armando, A., Carbone, R., Compagna, L., Cuellar, J., and Tobarra, L. (2008). Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-on for Google Apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering*, pages 1–10. ACM.

[8] Armando, A. and Compagna, L. (2004). SATMC: a SAT-based model checker for security protocols. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04)*, volume 3229 of *Lecture Notes in Artificial Intelligence*, pages 730–733. Springer.

[9] Backes, M., Dreier, J., Kremer, S., and Künnemann, R. (2017). A Novel Approach for Reasoning about Liveness in Cryptographic Protocols and Its Application to Fair Exchange. In *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 76–91. IEEE.

[10] Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. MIT press.

[11] Banfield, J. M. (2016). *A Study of Information Security Awareness Program Effectiveness in Predicting End-User Security Behavior*. PhD thesis, Eastern Michigan University.

[12] Barthe, G., Grégoire, B., Heraud, S., and Zanella Béguelin, S. (2009). Formal Certification of ElGamal Encryption. In *Proceedings of the 8th International Workshop of Formal Aspects in Security and Trust (FAST08)*, volume 5491 of *Lecture Notes in Computer Science*, pages 1–19. Springer.

[13] Basin, D. A., Caleiro, C., Ramos, J., and Viganò, L. (2011). Distributed temporal logic for the analysis of security protocol models. *Theoretical Computer Science*, 412(31):4007–4043.

[14] Basin, D. A., Mödersheim, S., and Viganò, L. (2005). OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208.

[15] Basin, D. A., Radomirovic, S., and Schläpfer, M. (2015). A Complete Characterization of Secure Human-Server Communication. In *Proceedings of the IEEE 28th Computer Security Foundations Symposium*, pages 199–213. IEEE.

[16] Basin, D. A., Radomirovic, S., and Schmid, L. (2016). Modeling Human Errors in Security Protocols. In *Proceedings of the 29th IEEE Computer Security Foundations Symposium (CSF)*, pages 325–340. IEEE.

[17] BBC (uk) (2016). Forged rail tickets sold on 'dark web', BBC investigation reveals. https://bbc.co.uk/news/uk-england-37800623.

[18] Beckert, B. and Beuster, G. (2006). A Method for Formalizing, Analyzing, and Verifying Secure User Interfaces. In *Proceedings of the 8th International Conference*

on Formal Methods and Software Engineering, volume 4260 of *Lecture Notes in Computer Science*, pages 55–73. Springer.

[19] Bella, G. and Coles-Kemp, L. (2011). Internet Users' Security and Privacy While They Interact with Amazon. In *Proceedings of the IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 878–883. IEEE.

[20] Bella, G. and Coles-Kemp, L. (2012). Layered Analysis of Security Ceremonies. In *Proceedings of the 27th IFIP TC 11 Information Security and Privacy Conference*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 273–286. Springer.

[21] Bella, G., Curzon, P., and Lenzini, G. (2015). Service Security and Privacy as a Socio-Technical Problem. *IOS Journal of Computer Security*, 23(5):563–585.

[22] Bella, G., Giustolisi, R., and Lenzini, G. (2013). Socio-technical formal analysis of TLS certificate validation in modern browsers. In *Proceedings of the 11th Annual Conference on Privacy, Security and Trust*, pages 309–316. IEEE.

[23] Bella, G., Giustolisi, R., and Lenzini, G. (2018). Invalid Certificates in Modern Browsers: A Socio-Technical Analysis. *IOS Journal of Computer Security*, 26(4):509–541.

[24] Bella, G., Giustolisi, R., Lenzini, G., and Ryan, P. Y. (2017). Trustworthy Exams without Trusted Parties. *Computer Security*, 67:291–307.

[25] Bella, G., Renaud, K., Sempreboni, D., and Viganò, L. (2019). An Investigation into the "Beautification" of Security Ceremonies. In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE - Volume 2: SECRYPT*, pages 125–136. Scitepress Digital Library.

[26] Bella, G. and Viganò, L. (2015). Security is Beautiful. In *Proceedings of the 23rd International Workshop on Security Protocols (SPW'16)*, volume 9379 of *Lecture Notes in Computer Science*, pages 247–250. Springer.

[27] Bellare, M. and Rogaway, P. (2006). The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, page 409–426. Springer.

[28] Blanchet, B. (2001). An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, pages 82–96. IEEE.

[29] Blanchet, B. (2006). A Computationally Sound Mechanized Prover for Security Protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 140–154. IEEE.

[30] Blaze, M. (2004). Toward a Broader View of Security Protocols. In *Proceedings of the 12th International Conference on Security Protocols*, volume 3957 of *Lecture Notes in Computer Science*, pages 106–120. Springer.

[31] Blythe, J., Koppel, R., and Smith, S. W. (2013). Circumvention of Security: Good Users Do Bad Things. *IEEE Security and Privacy*, 11(5):80–83.

[32] Bowman, H., Faconti, G., and Massink, M. (2001). Towards integrated cognitive and interface analysis. *Electronic Notes in Theoretical Computer Science*, 43:97–112.

[33] Büchler, M., Oudinet, J., and Pretschner, A. (2011). Security Mutants for Property-Based Testing. In *Tests and Proofs*, volume 6706 of *Lecture Notes in Computer Science*, pages 69–77. Springer.

[34] Canetti, R. and Krawczyk, H. (2001). Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'01)*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer.

[35] Carlos, M. (2014). *Towards a Multidisciplinary Framework for the Design and Analysis of Security Ceremonies*. PhD thesis, Royal Holloway, University of London.

[36] Carlos, M. C., Martina, J. E., Price, G., and Custódio, R. F. (2012). A proposed framework for analysing security ceremonies. In *Proceedings of the International Conference on Security and Cryptography - Volume 1: SECRYPT, (ICETE 2012)*, pages 440–445. INSTICC, Scitepress Digital Library.

[37] Carlos, M. C., Martina, J. E., Price, G., and Custódio, R. F. (2013). An Updated Threat Model for Security Ceremonies. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1836–1843. ACM.

[38] Carritt, E. F. (1932). *What is Beauty?* Clarendon Press, London.

[39] Carver, R. (1993). Mutation-based testing of concurrent programs. In *Proceedings of IEEE International Test Conference (ITC)*, pages 845–853. IEEE.

[40] Castelfranchi, C. and Tan, Y.-H. (2001). *Trust and Deception in Virtual Societies*. Springer.

[41] Chen, J., Kanj, I. A., and Xia, G. (2005). Simplicity is Beauty: Improved Upper Bounds for Vertex Cover. *Manuscript communicated by email*.

[42] Cheval, V., Cortier, V., and Warinschi, B. (2017). Secure composition of PKIs with Public Key Protocols. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF)*, pages 144–158. IEEE.

[43] Clark, S., Goodspeed, T., Metzger, P., Wasserman, Z., Xu, K., and Blaze, M. (2011). Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-Way Radio System. In *Proceedings of the 20th USENIX Conference on Security*, pages 8–12. USENIX Association.

[44] Cooper, A. et al. (2004). *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*, volume 2. Sams Indianapolis.

[45] Cortier, V., Delaitre, J., and Delaune, S. (2007). Safely composing security protocols. In *Proceedings of the 27th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4855 of *Lecture Notes in Computer Science*, pages 352–363. Springer.

[46] Courtois, N., Nohl, K., and O'Neil, S. (2008). Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards. *IACR Cryptology ePrint Archive*, page 166.

[47] Cranor, L. F. and Garfinkel, S. (2005). *Security and Usability: Designing Secure Systems that People Can Use*. O'Reilly Media, Inc.

[48] Creese, S., Goldsmith, M., and Roscoe, B. (2003). The attacker in ubiquitous computing environments: formalising the threat model. In *Proceedings of the 1st International Workshop of Formal Aspects in Security and Trust (FAST03)*.

[49] Cremers, C. J. F. (2008). The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Proceedings of the 20th International Conference on Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418. Springer.

[50] Curzon, P., Rukšėnas, R., and Blandford, A. (2007). An approach to formal verification of human-computer interaction. *Formal Aspects of Computing*, 19(4):513–550.

[51] Dadeau, F., Héam, P.-C., Kheddam, R., Maatoug, G., and Rusinowitch, M. (2015). Model-based mutation testing from security protocols in hlpsl. *Software Testing, Verification and Reliability*, 25(5–7):684–711.

[52] Dalpiaz, F., Paja, E., and Giorgini, P. (2011). Security requirements engineering via commitments. In *1st Workshop on Socio-Technical Aspects in Security and Trust (STAST)*, pages 1–8. IEEE.

[53] de Koning Gans, G., Hoepman, J.-H., and Garcia, F. D. (2008). A practical attack on the MiFare Classic. In *International Conference on Smart Card Research and Advanced Applications*, volume 5189 of *Lecture Notes in Computer Science*, pages 267–282. Springer.

[54] De Souza, S. D. R. S., Maldonado, J. C., Fabbri, S. C. P. F., and De Souza, W. L. (1999). Mutation Testing Applied to Estelle Specifications. *Software Quality Journal*, 8(4):285–301.

[55] DeMillo, R. A., Lipton, R. J., and Sayward, F. G. (1979). Program Mutation: A New Approach to Program Testing. *Infotech State of the Art Report, Software Testing*.

[56] Dolev, D. and Yao, A. (1983). On the Security of Public Key Protocols. *IEEE Transactions on information theory*, 29(2):198–208.

[57] Dowling, B., Fischlin, M., Günther, F., and Stebila, D. (2015). A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, page 1197–1210. ACM.

[58] Ellison, C. M. (2007). Ceremony Design and Analysis. *IACR Cryptology ePrint Archive*, 399:1–17.

[59] Emery, F. E. and Trist, E. L. (1960). Socio-technical systems. In *Management Science Models and Techniques*, volume 2, page 83–97. Pergamon,Oxford, UK.

[60] Erickson, M. (2011). *Beautiful Mathematics*. The Mathematical Association of America.

[61] Escobar, S., Meadows, C., and Meseguer, J. (2009). *Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties*, page 1–50. Springer.

[62] Fabbri, S. C. P. F., Maldonado, J. C., Sugeta, T., and Masiero, P. C. (1999). Mutation testing applied to validate specifications based on statecharts. In *Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No.PR00443)*, pages 210–219.

[63] Fábrega, T., Javier, F., Herzog, J. C., and Guttman, J. D. (1999). Strand spaces: Proving security protocols correct. *Journal of computer security*, 7(2–3):191–230.

[64] Ferreira, A., Huynen, J.-L., Koenig, V., and Lenzini, G. (2014). A Conceptual Framework to Study Socio-Technical Security. In *Proceedings of the 2nd International Conference on Human Aspects of Information Security, Privacy, and Trust - Volume 8533*, Lecture Notes in Computer Science, page 318–329. Springer.

[65] Ferreira, A. and Teles, S. (2019). Persuasion: How phishing emails can influence users and bypass security measures. *International Journal of Human-Computer Studies*, 125:19–31.

[66] Garcia, F. D., Koning Gans, G., Muijrers, R., Rossum, P., Verdult, R., Schreur, R. W., and Jacobs, B. (2008). Dismantling MIFARE Classic. In *Proceedings of the*

13th European Symposium on Research in Computer Security: Computer Security, Lecture Notes in Computer Science, page 97–114. Springer.

[67] Gelernter, D. H. (1998). *Machine Beauty: Elegance and the Heart of Technology*. Perseus Books, L.L.C., 1st edition.

[68] Giustolisi, R. (2017). Free Rides in Denmark: Lessons from Improperly Generated Mobile Transport Tickets. In *Proceedings of the 22th Nordic Conference on Secure IT Systems*, volume 10674 of *Lecture Notes in Computer Science*, pages 159–174. Springer.

[69] Glynn, I. (2010). *Elegance in science: the beauty of simplicity*. Oxford University Press.

[70] Grassi, P. A., Fenton, J. L., Newton, E. M., Perlner, R. A., Regenscheid, A. R., Burr, W. E., and Richer, J. P. (2017). Nist special publication 800-63. https://pages.nist.gov/800-63-3/sp800-63b.html.

[71] Greitzer, F., Purl, J., Leong, Y. M., and Becker, D. E. S. (2018). SOFIT: Sociotechnical and Organizational Factors for Insider Threat. In *Proceedings of the 39th IEEE Security and Privacy Workshops (SPW)*, pages 197–206. IEEE.

[72] Groß, T. and Mödersheim, S. (2011). Vertical protocol composition. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium*, pages 235–250.

[73] Guttman, J. D. (2009). Cryptographic protocol composition via the authentication tests. In *Foundations of Software Science and Computational Structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 303–317. Springer.

[74] Hall, J. (2018). SplashData's Top 100 Worst Passwords of 2018. https://www.teamsid.com/splashdatas-top-100-worst-passwords-of-2018/.

[75] Hassenzahl, M. and Monk, A. (2010). The Inference of Perceived Usability From Beauty. *Human–Computer Interaction*, 25(3):235–260.

[76] Herley, C. (2009). So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*, pages 133–144. ACM.

[77] Hess, A. V., Mödersheim, S., and Brucker, A. D. (2018). Stateful Protocol Composition. In *Proceedings of the 23th European Symposium on Research in Computer Security*, volume 11098 of *Lecture Notes in Computer Science*, pages 427–446. Springer.

[78] Holzmann, G. (2011). *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional.

[79] Inglesant, P. G. and Sasse, M. A. (2010). The true cost of unusable password policies: Password use in the wild. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 383–392. ACM.

[80] iotaseed.io (2019). The IOTA Scam - A Malicious IOTA Seed Generator. https://iotaseed.io/iota-seed-generator-scam/.

[81] Jagatic, T. N., Johnson, N. A., Jakobsson, M., and Menczer, F. (2007). Social Phishing. *ACM*, 50(10):94–100.

[82] Jansen, W. A. (2011). Cloud hooks: Security and privacy issues in cloud computing. In *Proceedings of the 44th Hawaii International Conference on System Sciences*, pages 1–10. IEEE.

[83] Jia, Y. and Harman, M. (2011). An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering*, 37(5):649–678.

[84] Johansen, C. and Jøsang, A. (2015). Probabilistic Modelling of Humans in Security Ceremonies. In *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, volume 8872 of *Lecture Notes in Computer Science*, pages 277–292. Springer.

[85] Kacmaz, U., Herrmann, T., and Jelonek, M. (2020). SeeMe2BPMN: Extending the Socio-Technical Walkthrough with BPMN. In *International Conference on Human-Computer Interaction*, volume 12184 of *Lecture Notes in Computer Science*, pages 72–84. Springer.

[86] Kammüller, F. (2017). Formal Modeling and Analysis with Humans in Infrastructures for IoT Health Care Systems. In *Human Aspects of Information Security, Privacy and Trust*, Lecture Notes in Computer Science, pages 339–352. Springer.

[87] Kammüller, F. and Probst, C. W. (2017). Modeling and Verification of Insider Threats Using Logical Analysis. *IEEE Systems Journal*, 11(2):534–545.

[88] Karlof, C., Tygar, J. D., and Wagner, D. (2009). Conditioned-Safe Ceremonies and a User Study of an Application to Web Authentication. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM.

[89] Karvonen, K. (2000). The Beauty of Simplicity. In *Proceedings of the Conference on Universal Usability*, pages 85–90. ACM.

[90] Kecheng Liu (2008). Pervasive informatics in intelligent spaces for living and working. In *Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics*, volume 1, pages 18–19. IEEE.

[91] Kennedy, S. (2016). The pathway to security–mitigating user negligence. *Information and Computer Security*, 24(3):255–264.

[92] Kiayias, A., Zacharias, T., and Zhang, B. (2017). Ceremonies for End-to-End Verifiable Elections. In *Proceedings of the 20th IACR International Conference on Public-Key Cryptography*, volume 10175 of *Lecture Notes in Computer Science*, pages 305–334. Springer.

[93] Kumar, A., Saxena, N., Tsudik, G., and Uzun, E. (2009). A Comparative Study of Secure Device Pairing Methods. *Pervasive Mob. Comput.*, 5(6):734–749.

[94] Lampson, B. (2009). Usable Security: How to Get It. *ACM*, 52(11):25–27.

[95] Liu, K., Nakata, K., and Harty, C. (2010). Pervasive informatics: theory, practice and future directions. *Intelligent Buildings International*, 2(1):5–19.

[96] Lortz, V. B., Walker, J. R., Hegde, S. S., Kulkarni, A. A., and Tai, T.-Y. C. (2012). Device introduction and access control framework. US Patent 8,146,142.

[97] Loveland, D. W. (2016). *Automated Theorem Proving: A Logical Basis*. Elsevier.

[98] Lowe, G. (1996). Breaking and Fixing the Needham-Shroeder Public-Key Protocol Using FDR. In *Proceedings of the 2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, page 147–166. Springer.

[99] Mannan, M. and van Oorschot, P. C. (2011). Leveraging Personal Devices for Stronger Password Authentication from Untrusted Computers. *Journal of Computer Security*, 19(4):703–750.

[100] Martimiano, T. and Martina, J. E. (2018). Daemones Non Operantur Nisi Per Artem (Daemons Do Not Operate Save Through Trickery: Human Tailored Threat Models for Formal Verification of Fail-Safe Security Ceremonies). In *Proceedings of the 25th International Workshop on Security Protocols (SPW'18)*, volume 11286 of *Lecture Notes in Computer Science*, pages 96–105. Springer.

[101] Martimiano, T., Martina, J. E., Olembo, M. M., and Carlos, M. C. (2014). Modelling User Devices in Security Ceremonies. In *Proceedings of the Workshop on Socio-Technical Aspects in Security and Trust*, pages 16–23.

[102] Martina, J. and Carlos, M. (2008). Why should we analyze security ceremonies. *Applications of Logic in Computer Security. The 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*.

[103] Martina, J. E., de Souza, T. C. S., and Custodio, R. F. (2009). Ceremonies Formal Analysis in PKI's Context. In *Proceedings of the International Conference on Computational Science and Engineering - Volume 03*, page 392–398. IEEE.

[104] Martina, J. E., Santos, E., Carlos, M. C., Price, G., and Custódio, R. F. (2015). An Adaptive Threat Model for Security Ceremonies. *International Journal of Information Security*, 14(2):103–121.

[105] Masci, P., Curzon, P., Blandford, A., and Furniss, D. (2011). Modelling Distributed Cognition Systems in PVS. *ECEASST*, 45.

[106] Masci, P., Rukšėnas, R., Oladimeji, P., Cauchi, A., Gimblett, A., Li, Y., Curzon, P., and Thimbleby, H. (2015). The benefits of formalising design guidelines: a case study on the predictability of drug infusion pumps. *Innovations in Systems and Software Engineering*, 11(2):73–93.

[107] McMillan, R. (2017). The Man Who Wrote Those Password Rules Has a New Tip. https://www.wsj.com/articles/the-man-who-wrote-those-password-rules-has-a-new-tip-n3v-r-m1-d-1502124118.

[108] Meier, S., Schmidt, B., Cremers, C., and Basin, D. A. (2013). The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Proceedings of the 25th International Conference on Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer.

[109] Millen, J., Clark, S., and Freedman, S. (1987). The Interrogator: Protocol Security Analysis. *IEEE Transactions on Software Engineering*, 13(2):274–288.

[110] Mitnick, K. D. and Simon, W. L. (2002). *The Art of Deception: Controlling the Human Element of Security.* John Wiley and Sons Ltd.

[111] Model, B. P. (2011). Notation (BPMN) version 2.0. *OMG Specification, Object Management Group*, pages 22–31.

[112] Mödersheim, S. and Viganò, L. (2009a). Secure Pseudonymous Channels. In *Proceedings of the 14th European Conference on Research in Computer Security*, volume 5789 of *Lecture Notes in Computer Science*, pages 337–354. Springer.

[113] Mödersheim, S. and Viganò, L. (2009b). *The Open-Source Fixed-Point Model Checker for Symbolic Analysis of Security Protocols*, volume 5705 of *Lecture Notes in Computer Science*, pages 166–194. Springer.

[114] Mödersheim, S. and Viganò, L. (2014). Sufficient conditions for vertical composition of security protocols. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS)*, pages 435–446. ACM.

[115] Murnane, T. and Reed, K. (2001). On the effectiveness of mutation analysis as a black box testing technique. In *Proceedings 2001 Australian Software Engineering Conference*, pages 12–20. IEEE.

[116] Nass, C., Isbister, K., and Lee, E.-J. (2001). *Truth is Beauty: Researching Embodied Conversational Agents*, pages 374–402. MIT Press.

[117] Needham, R. M. and Schroeder, M. D. (1978). Using Encryption for Authentication in Large Networks of Computers. *ACM*, 21(12):993–999.

[118] Nilsson, R., Offutt, J., and Mellin, J. (2006). Test case generation for mutation-based testing of timeliness. *Electronic Notes in Theoretical Computer Science*, 164(4):97–114.

[119] Olivo, C. K., Santin, A. O., and Oliveira, L. S. (2013). Obtaining the Threat Model for E-Mail Phishing. *Appl. Soft Comput.*, 13(12):4841–4848.

[120] Paja, E., Dalpiaz, F., and Giorgini, P. (2013). Designing Secure Socio-Technical Systems with STS-ml. In *Proceedings of the 6th International i* Workshop*, volume 978, pages 79–84. CEUR.

[121] Pancake, C. (2001). The Ubiquitous Beauty of User-Aware Software. *ACM*, 44(3):130–130.

[122] Parkin, S., van Moorsel, A., Inglesant, P., and Sasse, M. A. (2010). A stealth approach to usable security: Helping it security managers to identify workable security solutions. In *Proceedings of the 2010 New Security Paradigms Workshop*, pages 33–50. ACM.

[123] Parr, T. (2013). *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf.

[124] Paulson, L. C. (1998). The Inductive Approach to Verifying Cryptographic Protocols. *Journal of computer security*, 6(1–2):85–128.

[125] Pavlovic, D. and Meadows, C. (2012). Actor-Network Procedures (Extended Abstract). In *Proceedings of the 8th International Conference on Distributed Computing and Internet Technology (ICDCIT'12)*, volume 7154 of *Lecture Notes in Computer Science*, pages 7–26. Springer.

[126] Peppa Pig (2010). Peppa Pig, Series 3, Episode 38, "The Secret Club". https://youtu.be/WTeDMMl0ncI.

[127] Peroli, M., De Meo, F., Viganò, L., and Guardini, D. (2018). MobSTer: A Model-Based Security Testing Framework for Web Applications. *Software Testing, Verification & Reliability*, 28(8).

[128] Portanova, M. S. (1975). Music is beauty. *The Black Perspective in Music*, 3(2):196–198.

[129] Probst, C. W., Kammüller, F., and Hansen, R. R. (2016). *Formal Modelling and Analysis of Socio-Technical Systems*, pages 54–73. Lecture Notes in Computer Science. Springer.

[130] Radke, K. and Boyd, C. (2017). Security Proofs for Protocols Involving Humans. *The Computer Journal*, 60(4):527–540.

[131] Radke, K., Boyd, C., Nieto, J. M. G., and Brereton, M. (2011). Ceremony Analysis: Strengths and Weaknesses. In *Proceedings of the 26th IFIP International Information Security Conference*, volume 354 of *IFIP Advances in Information and Communication Technology*, pages 104–115. Springer.

[132] Reber, R., Schwarz, N., and Winkielman, P. (2004). Processing Fluency and Aesthetic Pleasure: Is Beauty in the Perceiver's Processing Experience? *Personality and Social Psychology Review*, 8(4):364–382.

[133] Rukšėnas, R., Curzon, P., and Blandford, A. (2008). Modelling and analysing cognitive causes of security breaches. *Innovations in Systems and Software Engineering*, 4(2):143–160.

[134] Rushby, J. (2001). Analyzing cockpit interfaces using formal methods. *Electronic Notes in Theoretical Computer Science*, 43:1–14.

[135] Russell, B. (1956). *The Autobiography of Bertrand Russell*. George Allen & Unwin.

[136] Sabbagh, B. A. and Kowalski, S. (2015). A Socio-technical Framework for Threat Modeling a Software Supply Chain. *IEEE Security Privacy*, 13(4):30–39.

[137] Salnitri, M. and Giorgini, P. (2014). Modeling and verification of ATM security policies with SecBPMN. In *2014 International Conference on High Performance Computing Simulation (HPCS)*, pages 588–591. IEEE.

[138] Salnitri, M. and Giorgini, P. (2014). Transforming Socio-Technical Security Requirements in SecBPMN Security Policies. In *Proceedings of the 7th International i^* Workshop*, volume 1157. CEUR.

[139] Salnitri, M., Paja, E., Giorgini, P., et al. (2015). From Socio-Technical Requirements to Technical Security Design: an STS-based Framework. Technical report, DISI, University of Trento.

[140] Saunders, M. N. (2011). *Research Methods for Business Students, 7/e*. Pearson Education.

[141] Schechter, S., Brush, A. B., and Egelman, S. (2009). It's No Secret. Measuring the Security and Reliability of Authentication via "Secret" Questions. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pages 375–390. IEEE.

[142] Schmidt, B. (2012). *Formal analysis of key exchange protocols and physical protocols*. PhD thesis, ETH Zurich.

[143] Schmidt, B., Meier, S., Cremers, C., and Basin, D. A. (2012). Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium*, pages 78–94. IEEE.

[144] Sciarretta, G., Carbone, R., Ranise, S., and Viganò, L. (2018). Design, Formal Specification and Analysis of Multi-Factor Authentication Solutions with a Single Sign-On Experience. In *Proceedings of the 7th International Conference on Principles of Security and Trust (POST'18)*, volume 10804 of *Lecture Notes in Computer Science*, pages 188–213. Springer.

[145] Semančík, R. (2007). Basic properties of the persona model. *Computing and Informatics*, 26(2):105–121.

[146] Sempreboni, D., Bella, G., Giustolisi, R., and Viganò, L. (2019a). The Danish Mobilpendlerkort Tamarin code. https://www.dropbox.com/s/o93hq0bh75lpxqk/mobilpendlerkort.spthy?dl=0.

[147] Sempreboni, D., Bella, G., Giustolisi, R., and Viganò, L. (2019b). What Are the Threats?(Charting the Threat Models of Security Ceremonies). In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE - Volume 2: SECRYPT*, pages 161–172. Scitepress Digital Library.

[148] Sempreboni, D. and Viganò, L. (2018). May I Mine Your Mind? In *Proceedings of the 2nd Re-Coding Black Mirror workshop, Companion of The Web Conference (WWW)*, pages 1573–1576. ACM.

[149] Sempreboni, D. and Viganò, L. (2019a). Privacy, Security and Trust in the Internet of Neurons. In *Proceedings of the 3rd Re-Coding Black Mirror workshop, Companion of the Data Protection and Artificial Intelligence (CPDP)*. arxiv.org/pdf/1807.06077.

[150] Sempreboni, D. and Viganò, L. (2019b). Smart Humans... WannaDie? In *Proceedings of the 3rd Re-Coding Black Mirror workshop, Companion of the Data Protection and Artificial Intelligence (CPDP)*. arxiv.org/abs/1812.05834.

[151] Sempreboni, D. and Viganò, L. (2020). X-Men: A Mutation-Based Approach for the Formal Analysis of Security Ceremonies. In *Proceedings of the 5th IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 87–104. IEEE.

[152] Shahriar, H. and Zulkernine, M. (2008). MUSIC: Mutation-based SQL Injection Vulnerability Checking. In *2008 The Eighth International Conference on Quality Software*, pages 77–86. IEEE.

[153] Shahriar, H. and Zulkernine, M. (2008). Mutation-based testing of buffer overflow vulnerabilities. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 979–984. IEEE.

[154] Shahriar, H. and Zulkernine, M. (2009). MUTEC: Mutation-based testing of Cross Site Scripting. In *2009 ICSE Workshop on Software Engineering for Secure Systems*, pages 47–53. IEEE.

[155] Song, D. X. (1999). Athena: a new efficient automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 192–202. IEEE.

[156] Soukoreff, R. W. and MacKenzie, I. S. (2001). Measuring Errors in Text Entry Tasks: An Application of the Levenshtein String Distance Statistic. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, pages 319–320. ACM.

[157] Stajano, F. (2011). Pico: No more passwords! In *Security Protocols Workshop*, volume 7114 of *Lecture Notes in Computer Science*, pages 49–81. Springer.

[158] Stajano, F. and Wilson, P. (2011). Understanding Scam Victims: Seven Principles for Systems Security. *ACM*, 54(3):70–75.

[159] Subashini, K. and Sumithra, G. (2014). Secure multimodal mobile authentication using one time password. In *Proceedings of the 2nd International Conference on Current Trends In Engineering and Technology - ICCTET*, pages 151–155. IEEE.

[160] Tamarin (2020a). The Tamarin Github code. https://tamarin-prover.github.io.

[161] Tamarin (2020b). The Tamarin User Manual. https://tamarin-prover.github.io.

[162] Tatarkiewicz, W. (2006). *History of Aesthetics: Edited by J. Harrell, C. Barrett and D. Petsch.* A&C Black.

[163] TfL (Transport for London). Card Clash. https://tfl.gov.uk/fares-and-payments/oyster/using-oyster/card-clash.

[164] TfL (Transport for London). Incomplete Journeys. https://tfl.gov.uk/fares-and-payments/oyster/using-oyster/incomplete-journeys.

[165] TfL (Transport for London). TfL Transparency Strategy. https://tfl.gov.uk/corporate/publications-and-reports/oyster-card.

[166] Thayer Fabrega, F., Herzog, J., and Guttman, J. (1998). Strand spaces: why is a security protocol correct? In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 160–171. IEEE.

[167] Thomas, D. and Moorby, P. (2008). *The Verilog Hardware Description Language.* Springer.

[168] Tractinsky, N., Katz, A., and Ikar, D. (2000). What is beautiful is usable. *Interacting with Computers*, 13(2):127–145.

[169] Tuch, A. N., Roth, S. P., Hornbæk, K., Opwis, K., and Bargas-Avila, J. A. (2012). Is beautiful really usable? Toward understanding the relation between usability, aesthetics, and affect in HCI. *Computers in Human Behavior*, 28(5):1596 – 1607.

[170] Turuani, M. (2006). The CL-Atse Protocol Analyser. In *Term Rewriting and Applications*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286. Springer.

[171] van Schaik, P., Jeske, D., Onibokun, J., Coventry, L., Jansen, J., and Kusev, P. (2017). Risk Perceptions of Cyber-Security and Precautionary Behaviour. *Comput. Hum. Behav.*, 75(C):547–559.

[172] Viganò, L. (2013). The SPaCIoS Project: Secure Provision and Consumption in the Internet of Services. In *Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation*, pages 497–498. IEEE.

[173] Viganò, L. and Sempreboni, D. (2018). Gnirut: The Trouble With Being Born Human In An Autonomous World. In *Proceedings of the 2nd Re-Coding Black Mirror workshop, Companion of The Web Conference (WWW)*, pages 1567–1571. ACM.

[174] Viganò, L. and Sempreboni, D. (2019). Schrödinger's Man. In *Proceedings of the 3rd Re-Coding Black Mirror workshop, Companion of the Data Protection and Artificial Intelligence (CPDP)*. arxiv.org/abs/1812.05839.

[175] White, S. A. (2004). Introduction to BPMN. *IBM Cooperation*.

[176] Wing, J. M. (1990). A Specifier's Introduction to Formal Methods. *Computer*, 23(9):8–23.

[177] Wu, P. P.-Y., Fookes, C., Pitchforth, J., and Mengersen, K. (2015). A Framework for Model Integration and Holistic Modelling of Socio-Technical Systems. *Decision Support Systems*, 71:14–27.

[178] X-Men (2020). X-Men: A Mutation-Based Approach for the Formal Analysis of Security Ceremonies. https://sempreboni.github.io/X-Men/.

[179] Yildirim, E. (2016). The Importance of Information Security Awareness for the Success of Business Enterprises. In *Advances in Human Factors in Cybersecurity*, volume 501 of *Advances in Intelligent Systems and Computing*, pages 211–222. Springer.

[180] Zhang, Y., Monrose, F., and Reiter, M. K. (2010). The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, page 176–186. ACM.

Appendix A

Tamarin files

A.1 The Needham-Schroeder Public Key Protocol

Here the Tamarin code for the Needham-Schroeder Public Key Protocol. The model was written bys Simon Meier and is available on the Tamarin distribution GitHub page [160].

```
theory NSPK3
begin

builtins: asymmetric-encryption

/*
   Protocol:    The classic three message version of the
                flawed Needham-Schroeder Public Key
                Protocol
   Modeler:     Simon Meier
   Date:        September 2012
```

```
 */

// Public key infrastructure
rule Register_pk:
  [ Fr(~ltkA) ]
  -->
  [ !Ltk($A, ~ltkA), !Pk($A, pk(~ltkA)), Out(pk(~ltkA)) ]

rule Reveal_ltk:
  [ !Ltk(A, ltkA) ] --[ RevLtk(A)    ]-> [ Out(ltkA) ]

  protocol NSPK3 {
    1. I -> R: {'1',ni,I}pk(R)
    2. I <- R: {'2',ni,nr}pk(I)
    3. I -> R: {'3',nr}pk(R)
  }
*/

rule I_1:
  let m1 = aenc{'1', ~ni, $I}pkR
  in
    [ Fr(~ni)
    , !Pk($R, pkR)
    ]
  --[ OUT_I_1(m1)
    ]->
```

```
    [ Out( m1 )
    , St_I_1($I, $R, ~ni)
    ]

rule R_1:
  let m1 = aenc{'1', ni, I}pk(ltkR)
      m2 = aenc{'2', ni, ~nr}pkI
  in
    [ !Ltk($R, ltkR)
    , In( m1 )
    , !Pk(I, pkI)
    , Fr(~nr)
    ]
  --[ IN_R_1_ni( ni, m1 )
    , OUT_R_1( m2 )
    , Running(I, $R, <'init',ni,~nr>)
    ]->
    [ Out( m2 )
    , St_R_1($R, I, ni, ~nr)
    ]

rule I_2:
  let m2 = aenc{'2', ni, nr}pk(ltkI)
      m3 = aenc{'3', nr}pkR
  in
    [ St_I_1(I, R, ni)
```

```
    , !Ltk(I, ltkI)
    , In( m2 )
    , !Pk(R, pkR)
    ]
  --[ IN_I_2_nr( nr, m2)
    , Commit (I, R, <'init',ni,nr>)
    , Running(R, I, <'resp',ni,nr>)
    ]->
    [ Out( m3 )
    , Secret(I,R,nr)
    , Secret(I,R,ni)
    ]


rule R_2:
    [ St_R_1(R, I, ni, nr)
    , !Ltk(R, ltkR)
    , In( aenc{'3', nr}pk(ltkR) )
    ]
  --[ Commit (R, I, <'resp',ni,nr>)
    ]->
    [ Secret(R,I,nr)
    , Secret(R,I,ni)
    ]


rule Secrecy_claim:
  [ Secret(A, B, m) ] --[ Secret(A, B, m) ]-> []
```

```
lemma types [sources]:
  " (All ni m1 #i.
       IN_R_1_ni( ni, m1) @ i
       ==>
       ( (Ex #j. KU(ni) @ j & j < i)
       | (Ex #j. OUT_I_1( m1 ) @ j)
       )
    )
  & (All nr m2 #i.
       IN_I_2_nr( nr, m2) @ i
       ==>
       ( (Ex #j. KU(nr) @ j & j < i)
       | (Ex #j. OUT_R_1( m2 ) @ j)
       )
    )
  "

/* Nonce secrecy from the perspective of both
the initiator and the responder.*/
lemma nonce_secrecy:
  " /* It cannot be that */
    not(
        Ex A B s #i.
          /* somebody claims to have setup a shared
          secret,*/
```

```
          Secret(A, B, s) @ i
          /* but the adversary knows it */
        & (Ex #j. K(s) @ j)
          /* without having performed a long-term key
          reveal.*/
        & not (Ex #r. RevLtk(A) @ r)
        & not (Ex #r. RevLtk(B) @ r)
       )"

// Injective agreement from the perspective of both the
initiator and the responder.
lemma injective_agree:
  " /* Whenever somebody commits to running a session,
   then*/
    All actor peer params #i.
        Commit(actor, peer, params) @ i
      ==>
        /* there is somebody running a session
        with the same parameters */
          (Ex #j. Running(actor, peer, params) @ j & j < i
            /* and there is no other commit on
            the same parameters */
            & not(Ex actor2 peer2 #i2.
                    Commit(actor2, peer2, params) @ i2
                    & not(#i = #i2)
                 )
```

```
          )
        /* or the adversary perform a long-term key reveal
        on actor or peer */
        | (Ex #r. RevLtk(actor) @ r)
        | (Ex #r. RevLtk(peer)  @ r)
  "

/* Consistency check: ensure that secrets can be shared
between honest agents.*/
lemma session_key_setup_possible:
  exists-trace
  " /* It is possible that */
    Ex A B s #i.
      /* somebody claims to have setup a shared secret, */
      Secret(A, B, s) @ i
      /* without the adversary having performed a
      long-term key reveal. */
    & not (Ex #r. RevLtk(A) @ r)
    & not (Ex #r. RevLtk(B) @ r)
  "
end
```

Listing A.1 The classic version of the Needham-Schroeder Public Key Protocol

A.2 The Danish Mobilpendlerkort

Here the Tamarin code for the Danish Mobilpendlerkort Ceremony. This code is original and it is also available online [146].

```
theory Mobilpendlerkort
begin
/*
Ceremony: Mobilpendlerkort Ticket inspection

Roles:
H ticket holder
I ticket inspector
P holder's phone/app
S inspector's scanner
Z DSB server
ATP attacking third party

Protocol in Alice&Bob notation:
A0. H     : knows(creditcard, phone)
A1. I     : knows(watermark, background)
A2. I     : knows (current_zone, current_date)
A3. S&Z   : know (ver_key) the QR code scanner stores the
            correct verification key

1.    H -> P    : details:phone_number, H, travelling_dates
                  ,travel_zones, credit_card
```

```
2.    P -> Z    : details //Z creates s1:watermark,
                  background, font, SN, H, travelling dates
                  ,travel_zone; and  sign_qr:sign(sn,
                  travelling_dates, travel_zones)
3.    Z -> S    : sn
4.    Z -> P    : s1, sign_qr
5.    P -> H    : s1, sign_qr
6.    H -> I    : s1, sign_qr
    *I vchecks that watermark, background, and font on S1
    match ??, I vchecks that the travel zones contain
    the current zone, I vchecks that starting date<current
    date<ending date
7.    I -> S    : sign_qr
    *S checks signature and travelling_dates, and valid sn
8.    S -> I      : OK
*/

/* built-ins */
builtins: signing

/* secure channel (i.e. confindential and authenticated) */
rule ChanOut_S:
  [Out_S($A,$B,xn,x)]
  --[ChanOut_S($A,$B,xn,x)]->
  [Sec($A,$B,xn,x)]
```

```
rule ChanIn_S:
  [Sec($A,$B,xn,x)]
  --[ChanIn_S($A,$B,xn,x)]->
  [In_S($A,$B,xn,x)]

/* setup */

/* the attacker should be able to get a CC */
rule ccgeneration:
  [Fr(~cc)]
  --[Hcc($H,~cc)]->
  [!CC($H,~cc)]

rule newpkey:
  [Fr(~skey)]
  --[Key($Z, ~skey)]->
  [!Pk($Z, pk(~skey)), !Seckey(~skey)]

rule setup:
     let details = <$phone, $H, $date, $zone> in
     [!CC($H, ~cc), !Pk($Z, vkey), !Seckey(~skey)]
     --[H($H), Roles($P,$Z,$S,$I)]->
     [!HK(details), !HK(~cc), Hst0($H, 'H_0',
     <details, ~cc>),
      !IK('wm', 'bg', $date, $zone), !ZK0(~skey)]
```

```
/*the human types the details into their phones*/
rule H_1:
     let details = <$phone, $H, $date, $zone> in
     [Hst0($H, 'H_0', <details, ~cc>)]
     --[SendCC(~cc)]->
     [Out_S($H, $P, 'details', <details, ~cc>), !Hst1($H,
     'H_1', <details, ~cc>)]

/* the phone forwards the details to the DSB server */
rule P_2:
     let details = <$phone, $H, $date, $zone> in
     [In_S($H, $P, 'details', <details, ~cc>)]
     -->
     [Pst2($P, 'P_2', details), Out_S($P, $Z, 'detailsPZ',
     <details, ~cc>)]

/* the server stores the sn for the scanner and sends
the ticket to the phone*/
rule Z_34:
     let details  = <$phone, $H, $date, $zone>
       s1   = <$H, $date, $zone, ~sn, 'wm', 'bg'>
       signed_qr  = sign(<$date, $zone, ~sn>, ~skey)
     in

     [In_S($P,$Z,'detailsPZ', <details, ~cc>), Fr(~sn),
     !Pk($Z, vkey), !ZK0(~skey)]
```

```
    --[Zassign(~sn, ~cc)]->
    [Zst2($Z, 'Z_34', 'make_ticket', <details, ~cc, ~sn,
    signed_qr>), !StoreSn($Z, $S, 'store_ticket', ~sn),
     Out_S($Z, $P, 'make_ticketZP', <s1, signed_qr>)]

/* P gets the ticket from the server */
rule P_5get:
    let details  = <$phone, $H, $date, $zone>
      s1   = <$H, $date, $zone, ~sn, 'wm', 'bg'>
    in
    [Pst2($P, 'P_2', details), In_S($Z, $P,
    'make_ticketZP', <s1, signed_qr>)]
    -->
    [!Pstore($P, 'P_5', <s1, signed_qr>)]

/* P leaks the ticket to ATP */
rule Pleaks:
    [!Pstore($P, 'P_5',<s1, signed_qr>)]
    --[PleakstoATP('ticket',signed_qr)]->
    [Out_S($P, $ATP, 'leak_ticket', signed_qr)]

/* On demand, P shows the right ticket to H*/
rule P_5out:
    [!Pstore($P, 'P_5', <s1, signed_qr>)]
    -->
    [Out_S($P, $H, 'fw_ticket', <s1, signed_qr>)]
```

```
rule ATPgets:
     [In_S($P, $ATP, 'leak_ticket', signed_qr)]
     --[ATPKnow($P, signed_qr)]->
     [!ATPK($P,  signed_qr)]

rule ATPsends:
     [!ATPK($P,  signed_qr)]
     -->
     [Out_S($ATPP, $PP, 'fake_ticket', signed_qr)]

/* P gets fake ticket from ATP*/
rule Pfw:
     [In_S($ATP, $P, 'fake_ticket', signed_qr_atp),
     !Pstore($P, 'P_5', <s1, signed_qr>)]
     --[Pgetsfake('ticket', s1,signed_qr_atp)]->
     [Out_S($P, $H, 'fw_ticket',<s1,signed_qr_atp>)]

/* note that Human can't parse signed_qr */
rule H_6:
     let details  = <$phone, $H, $date, $zone>
       s1   = <$H, $date, $zone, ~sn, 'wm', 'bg'>
     in
     [!Hst1($H, 'H_1', <details, ~cc>), In_S($P, $H,
     'fw_ticket', <s1, signed_qr>)]
     -->
```

```
      [Hst2($H, 'H_65', s1), Out_S($H, $I, 'show_ticket',
       <s1, signed_qr>)]

rule I_7:
      let s1   = <$H, $date, $zone, ~sn, 'wm', 'bg'>
      in
      [In_S($H, $I, 'show_ticket', <s1, signed_qr>),
      !IK('wm', 'bg', $date, $zone)]
      -->
      [Out_S($I, $S, 'switch_screen', <$H,signed_qr>),
      Ist0($I, 'I_7', $H, $S, $date, $zone)]

rule S_8:
      [!StoreSn($Z, $S, 'store_ticket',  ~sn), In_S($I, $S,
      'switch_screen', <$H, signed_qr>), !Pk($Z, vkey),
      Ist0($I, 'I_7', $H, $S, $date, $zone)]
      --[Eq(verify(signed_qr, <$date, $zone, ~sn>, vkey)
      , true), OK($H,~sn)]->
      []

restriction Equality:
  "All x y #i. Eq(x,y) @ i ==> x = y"

/* just one server and one public key pair*/
restriction singlekey:
  "All z1 z2 k1 k2 #i #j. Key(z1, k1)@i
```

```
  & Key(z2, k2)@j ==> z1=z2 & k1=k2"

/* lemmas claims */

/* Functionality Lemma */

lemma functional1: exists-trace
  "(All x y #i. Eq(x,y) @ i ==> x = y) &
  (Ex #k s m . OK(s,m) @k )"

/*if the scanner says OK for a sn, than the human sent
their cc and the server assigned that sn to that cc,
in other words someone paid for the ticket */
lemma functional2:
  "All hx s #k. OK(hx,s)@k ==>
  (Ex c #i #j. SendCC(c)@i & Zassign(s,c)@j & i<j  & j<k)"

/* the same ticket may be inspected moret than once */
lemma  functional3: exists-trace
  "Ex h1 s1 #i #j.  OK(h1,s1)@i & OK(h1,s1)@j & i<j"

/* two different humans cannot ride with the same ticket */
lemma oneticketpertraveller:
  "All h1 h2 s #i #j. OK(h1,s)@i
    & OK(h2,s)@j & i<j ==> h1=h2"
```

```
/* without malicious P and ATP no attack */
lemma correctness:
"(All s1  signed_qr #i.
    Pgetsfake('ticket', s1,signed_qr)@i ==> F) ==>
    (All h1 h2 s #j #k. OK(h1,s)@j
      & OK(h2,s)@k & j<k ==> h1=h2)"

end
```

Listing A.2 The full code of the Danish Mobilpendlerkort ceremony

A.3 The Oyster Ceremony

Here the Tamarin code the Oyster Ceremony. This code is original and it is also available online [178].

```
/****MODEL****/
theory Oyster

begin

/* Channel rules */
rule ChanSndS:
  [SndS($A,$B,xn,x)]
  --[ChanSndS($A,$B,xn,x)]->
  [!Sec($A,$B,xn,x)]

rule ChanRcvS:
  [!Sec($A,$B,xn,x)]
  --[ChanRcvS($A,$B,xn,x)]->
  [RcvS($A,$B,xn,x)]

/****RULES****/

builtins: asymmetric-encryption

functions: bal/1
```

```
rule humansetup:
  [ Fr(~gid)
    , Fr(~goid)
  ]
  --[OnlyOnce()
  , Neq($oyster,$ccard)]->
  [ !Type($Human,'card',$oyster)
    , !Type($Human,'card',$ccard)
    , !Type($Human,'balance',bal($oyster))
    , !Type($Human,'balance',bal($ccard))
    , !Type($GateIn,'gid',~gid)
    , !Type($GateIn,'goid',~goid)
    , !Wallet($oyster,$ccard)
    , !GateID($GateIn,~gid)
    , !GateID($GateOut,~goid)
    ]


/* setup */
rule Setup:
    [ !Wallet($oyster,$ccard)
    , !GateID($GateIn,~gid)
    , !GateID($GateOut,~goid)
    ]
    --[ Setup($Human)
    , Roles($Human,$GateIn,$GateOut)
    , GateIn($Human,$GateIn)
```

```
    , GateOut($Human,$GateOut)
  ]->
    [ State($GateIn,'1',<~gid>)
    , State($GateOut,'1',<~goid>)
    , State($Human,'1',<$oyster,$ccard,bal($oyster)
    ,bal($ccard)>)
    ]

rule H_1:
  [ State($Human,'1',<$oyster,$ccard,bal($oyster)
  ,bal($ccard)>)]
  --[H()
  , Send($Human,'card',$oyster)
  , To($GateIn)
  ]->
  [ State($Human,'2',<$oyster,$ccard,bal($oyster)
  ,bal($ccard)>)
  , SndS($Human,$GateIn,'card',$oyster)
  ]

rule GateIn_1:
  [ State($GateIn,'1',<~gid>)
  , RcvS($Human,$GateIn,'card',$oyster)
  ]
  --[Receive($GateIn,$Human,$oyster)
  , CommitGid($GateIn,$Human,~gid)]->
```

```
  [ State($GateIn,'2',<~gid,$Human,$oyster>)
  , SndS($GateIn,$Human,<'card','gid'>,<$oyster,~gid>)
  ]

rule H_2:
  [ State($Human,'2',<$oyster,$ccard,bal($oyster)
  ,bal($ccard)>)
  , RcvS($GateIn,$Human,<'card','gid'>,<$oyster,~gid>)
  ]
  --[H()
  , Receive($Human,$GateIn,~gid)
  , Send($Human,'card',$oyster)
  , Send($Human,'balance',bal($oyster))
  , Send($Human,'gid',~gid)
  , To($GateOut)]->
  [ State($Human,'3',<$oyster,$ccard,bal($oyster)
  ,bal($ccard),~gid>)
  , SndS($Human,$GateOut,<'card','balance','gid'>,<$oyster
  ,bal($oyster),~gid>)
  ]

rule GateOut_1:
  [ State($GateOut,'1',<~goid>)
  , RcvS($Human,$GateOut,<'card','balance','gid'>,<$oyster
  ,bal($oyster),~gid>)
  ]
```

```
  --[ Receive($GateOut,$Human,$oyster)
  , Receive($GateOut,$Human,bal($oyster))
  , Commit($GateOut,$Human,'finish')]->
  [ State($GateOut,'2',<~goid,$oyster,bal($oyster),~gid>)
  , SndS($GateOut,$Human,<'card','balance','finish'>
  ,<$oyster,bal($oyster),'finish'>)
  ]

rule H_3:
  [ State($Human,'3',<$oyster,$ccard,bal($oyster)
  ,bal($ccard),~gid>)
  , RcvS($GateOut,$Human,<'card','balance','finish'>
  ,<$oyster,bal($oyster),'finish'>)
  ]
  --[H()
  , Hfin($Human,'card',$oyster)]->
  []

/****ENDOFRULES****/

restriction notSameRole:
  "All H1 H2 GIN1 GIN2 GOUT1 GOUT2 #i #j.
    Roles(H1,GIN1,GOUT1) @i & Roles(H2,GIN2,GOUT2) @j ==>
      not H1 = GIN1
    & not H1 = GIN2
    & not H1 = GOUT1
```

```
    & not H1 = GOUT2
    & not GIN1 = GOUT1
    & not GIN1 = GOUT2
    & H1 = H2
  "

restriction Inequality:
  "All x #i. Neq(x,x) @#i ==> F"

restriction OnlyOnce:
  "All #i #j. OnlyOnce()@#i & OnlyOnce()@#j ==> #i = #j"

lemma Same_Card: all-traces
  "(All H oyster #j. Hfin(H,'card',oyster) @j
  ==> (Ex #t. Send(H,'card',oyster) @ t & t<j)
    & not (Ex ccard #c. Send(H,'card',ccard) @c
    & not (ccard = oyster))
    )"

lemma Card_Clash: all-traces
  "(All H GateIn GateOut oyster gid #j #t.
      Receive(GateOut,H,oyster) @j
      & Commit(GateOut,H,'finish') @j
      & Receive(GateIn,H,oyster) @t
      & CommitGid(GateIn,H,gid) @t
      &  t<j & not (GateIn = GateOut)
```

```
        ==> not (Ex ccard #i #k.
          Receive(GateOut,H,ccard) @i
          & Commit(GateOut,H,'finish') @i
          & Receive(GateIn,H,ccard) @k
          & CommitGid(GateIn,H,gid) @k
          & k<i
          & not oyster = ccard))"

lemma Complete_Journey: all-traces
  "(All H oyster #j. Hfin(H,'card',oyster) @j
    ==> (Ex GateIn m #i. CommitGid(GateIn,H,m) @i & i<j))"

/* Functional lemma */

lemma functional: exists-trace
  "(All H1 H2 #i #j.
    Setup(H1) @i & Setup(H2) @j ==> #i = #j)
    & (Ex H oyster GateOut #k #n.
    Hfin(H,'card',oyster) @k
    & Commit(GateOut,'Human','finish') @n)"

end

/****ENDOFMODEL****/
```

Listing A.3 The full code of the Oyster Ceremony

A.4 The Single Sign-On ceremony

Here the Tamarin code for the Single Sign-On Ceremony. This code is original and it is also available online [178].

```
/****MODEL****/


theory SSO


begin


builtins: signing, asymmetric-encryption


rule send_secure:
  [SndS(A,B,m)]
  -->
  [Sec(A,B,m)]


rule receive_secure:
  [Sec(A,B,m)]
  -->
  [RcvS(A,B,m)]


rule send_dy:
  [SndDY(A,B,m)]
  -->
  [Out(<A,B,m>)]
```

```
rule receive_dy:
  [In(<A,B,m>)]
  -->
  [RcvDY(A,B,m)]

rule Register_pk:
  [ Fr(~x) ]
  -->
  [ !Ltk($nx, ~x), !Pk($nx, pk(~x)), Out(pk(~x))]

rule Register_pkNotReveal:
  [ Fr(~ltkIdP) ]
  -->
  [ !IdPLtk($IdP, ~ltkIdP), !PkIdP($IdP, pk(~ltkIdP))
  , Out(pk(~ltkIdP))
  ]

rule Reveal_ltk:
  [ !Ltk($A, ltkA) ]
  --[ RevLtk($A)]->
  [ Out(ltkA) ]

/****RULES****/

rule setup:
```

```
  [ Fr(~id_SP)]
  --[Roles($Client,$SP,$IdP)]->
  [ State($Client,'1',<$SP>)
  , State($SP,'1',<~id_SP>)
  , State($IdP,'1',<'init'>)]

rule C_1:
  let
    httpReq = aenc{'get', ~URI, $Client, 'nil'}pkSP
  in
  [ State($Client,'1',<$SP>)
  , Fr(~URI),!Pk($SP, pkSP)]
  --[OUT_C_1(httpReq)]->
  [ State($Client,'2',<$SP,~URI>)
  , SndDY($Client,$SP,httpReq)
  ]

rule SP_1:
  let
    httpReq = aenc{'get', URI, Client, 'nil'}pk(ltkSP)
    authReq = <id_SP, $SP>
    httpResp = sign{'code302', $IdP, authReq, URI
    , 'nil'}ltkSP
  in
  [ State($SP,'1',<id_SP>)
  , RcvDY(Client,$SP,httpReq)
```

```
  , !Ltk($SP, ltkSP)]
  --[IN_SP_1_uri(URI,httpReq)
  , OUT_SP_2(httpResp)
  ]->
  [ State($SP,'2',<id_SP,URI,Client>)
  , SndS($SP,Client,httpResp)
  ]

rule C_2:
  let
    authReq = <id_SP, SP>
    httpResp = sign{'code302', IdP, authReq, URI
    , 'nil'}ltkSP
    httpReq = aenc{'get', IdP, authReq, URI, 'nil'}pkIdP
  in
  [ State(Client,'2',<SP,URI>)
  , RcvDY(SP,Client,httpResp)
  , !Pk(SP,pkSP)
  , !PkIdP(IdP,pkIdP)]
  --[Running(SP,Client,<id_SP, Client, IdP, SP>)
  , IN_C_2_id_sp(id_SP,httpResp)
  , OUT_C_2(httpReq)
  ]->
  [ State(Client,'3',<SP,URI,IdP,id_SP>)
  , SndS(Client,IdP,httpReq)
  ]
```

```
rule IdP_1:
  let
    authReq = <id_SP, SP>
    httpReq = aenc{'get', $IdP, authReq, URI
    , 'nil'}pk(ltkIdP)
    msg = < id_SP, Client, $IdP, SP >
    AA = sign{ msg }ltkIdP
    respIdP = sign{'code200','nil',SP,AA,URI}ltkIdP
    httpRespIdP = aenc{respIdP}pkClient
  in
  [ State($IdP,'1',<'init'>),RcvS(Client,$IdP,httpReq)
  , !IdPLtk($IdP, ltkIdP)
  , !Pk(Client,pkClient)
  ]
  --[H()
  , IN_IdP_1_id_sp(id_SP,httpReq)
  ]->
  [SndS($IdP,Client,httpRespIdP)]

rule C_3:
  let
    msg = < id_SP, Client, IdP, SP >
    AA = sign{ msg }ltkIdP
    httpReq = aenc{'post', SP, 'nil', AA, URI}pkSP
    msgIdP = <'code200','nil',SP,AA,URI>
```

```
    respIdP = sign{msgIdP}ltkIdP
    httpRespIdP = aenc{respIdP}pkClient
  in
  [ State(Client,'3',<SP,URI,IdP,id_SP>)
  , RcvS(IdP,Client,httpRespIdP)
  , !Pk(SP,pkSP)
  , !PkIdP(IdP,pkIdP)
  , !Ltk(Client, ltkClient)
  ]
  --[Eq(verify(respIdP,msgIdP,pkIdP),true)]->
  [ State(Client,'4',<SP,URI,IdP,id_SP>)
  , SndDY(Client,SP,httpReq)]

rule SP_2:
  let
    msg = < id_SP, Client, IdP, SP >
    AA = sign{ msg }ltkIdP
    httpReq = aenc{'post', SP, 'nil', AA, URI}pk(ltkSP)
    httpResp = sign{'code200',URI,'resource'}ltkSP
  in
  [ State(SP,'2',<id_SP,URI,Client>)
  , RcvDY(Client,SP,httpReq)
  , !PkIdP(IdP,pkIdP)
  , !Ltk(SP, ltkSP)
  ]
  --[Commit(SP,Client,<id_SP, Client, IdP, SP>)
```

```
  , Eq(verify(AA,msg,pkIdP),true)
  ]->
  [ SndS(SP,Client,httpResp)]

rule C_4:
  let
    msgX = <'code200',URI,'resource'>
    httpResp = sign{msgX}ltkSP
  in
  [ State(Client,'4',<SP,URI,IdP,id_SP>)
  , RcvDY(SP,Client,httpResp),!Pk(SP,pkSP)]
  --[Receive(SP,<'code200',URI,'resource'>)
  , Eq(verify(httpResp,msgX,pkSP),true)
  ]->
  []



/****ENDOFRULES****/

restriction Equality:
  "All x y #i. Eq(x,y) @i ==> x = y"

lemma types [sources]:
  "(All uri m1 #i.
    IN_SP_1_uri(uri, m1) @i
    ==>
```

```
    ((Ex #j. KU(uri) @j & j<i)
    | (Ex #j. OUT_C_1(m1) @j)
    )
  )
  & (All id_sp m2 #i.
    IN_C_2_id_sp(id_sp, m2) @i
    ==>
    ((Ex #j. KU(id_sp) @j & j<i)
    | (Ex #j. OUT_SP_2(m2) @j)
    )
  )
  & (All id_sp m2 #i.
    IN_IdP_1_id_sp(id_sp, m2) @i
    ==>
    ((Ex #j. KU(id_sp) @j & j<i)
    | (Ex #j. OUT_C_2(m2) @j)
    )
  )"

/* Injective agreement from the perspective of both
the initiator and the responder.*/
lemma injective_agree:
  " /* Whenever somebody commits
    to running a session, then*/
    All actor peer params #i.
        Commit(actor, peer, params) @ i
```

```
      ==>
        /* there is somebody running a session with the
        same parameters */
          (Ex #j. Running(actor, peer, params) @ j & j < i
            /* and there is no other commit on the
            same parameters */
            & not(Ex actor2 peer2 #i2.
                    Commit(actor2, peer2, params) @ i2
                    & not(#i = #i2)
                  )
          )
        /* or the adversary perform a long-term key reveal
        on actor or peer */
        | (Ex #r. RevLtk(actor) @ r)
        | (Ex #r. RevLtk(peer)  @ r)
  "

lemma functional: exists-trace
  "(Ex SP URI #m.
    Receive(SP,<'code200',URI,'resource'>)@m)"

end

/****ENDOFMODEL****/
```

Listing A.4 The full code of the Single Sign-On Ceremony

A.5 The Ticket Inspection Ceremony

Here the Tamarin code for the Ticket Inspection Ceremony. This code is original and it will be made public online.

```
/****MODEL****/
theory CoachService

begin

rule send_secure:
  [SndS(A,B,m)]
  -->
  [Sec(A,B,m)]

rule receive_secure:
  [Sec(A,B,m)]
  -->
  [RcvS(A,B,m)]

builtins: signing, symmetric-encryption

/****RULES****/

rule setup:
  [Fr(~kS)]
  --[OnlyOnce()
```

```
  , Roles($Client,$S,$D)]->
  [ State($S,'1',<~kS,$D>)
  , State($D,'1',~kS)
  , State($Client,'1',<$S,$D>)
  ]

rule H_1:
  let
    journey = < ~date, ~dtime, $from, $to >
  in
  [ State($Client,'1',<$S,$D>)
  , Fr(~date)
  , Fr(~dtime)
  ]
  --[]->
  [ State($Client,'2',<$S,$D,~date,~dtime,$from,$to>)
  , SndS($Client,$S,journey)
  ]

rule S_1:
  let
    journey = < date, dtime, from, to >
    solution = < date, dtime, from, to, ~price>
  in
  [ State($S,'1',<~kS,$D>)
  , RcvS(Client,$S,journey)
```

```
  , Fr(~price)
  ]
  --[]->
  [ State($S,'2',<~kS, $D, Client, ~price, date, dtime
  , from, to>), SndS($S,Client,solution)
  ]

rule H_2:
  let
    solution = < date, dtime, from, to, price>
    confirmation = <price,'ok'>
  in
  [ State(Client,'2',<S,D,date,dtime,from,to>)
  , RcvS(S,Client,solution)]
  --[]->
  [ State(Client,'3',<S,D,date,dtime,from,to,price>)
  , SndS(Client,S,confirmation)
  ]

rule S_2:
  let
    confirmation = <price,'ok'>
    ticket = <Client,~tknumber,price,date,dtime,from,to>
    encTicket = senc{ticket}kS
    msg = <ticket,encTicket>
  in
```

```
  [ State(S,'2',<kS, D, Client, price, date, dtime
  , from, to>)
  , RcvS(Client,S,confirmation)
  , Fr(~tknumber)
  ]
  --[ValidTicket(S,Client,'tn',~tknumber,date)]->
  [ SndS(S,Client,msg)
  ]

rule H_3:
  let
    ticket = <Client, tknumber, price, date, dtime
    , from, to>
    encTicket = senc{ticket}kS
    msg = <ticket,encTicket>
  in
  [ State(Client,'3',<S, D, date, dtime, from, to, price>)
  , RcvS(S,Client,msg)
  ]
  --[]->
  [ State(Client,'4',<S, D, date, dtime, from, to, price
  , tknumber>) , SndS(Client,D,msg)
  ]

rule D_1:
  let
```

```
    tck = <encClient,encTKnumber,encPrice,encDate,encDTime
    ,encFrom,encTo>
    encTicket = senc{tck}~kS
    ticket = <Client,tknumber,price,date,dtime,from,to>
    msg = <ticket,encTicket>
    ack = <tknumber,date, 'ack', 'valid'>
  in
  [ State($D,'1',~kS)
  , RcvS(Client,$D,msg)
  ]
  --[Eq(tknumber,encTKnumber)
  , Eq(Client,encClient)
  , Eq(price,encPrice)
  /*This is the action removed during the mutation*/
  , Eq(date,encDate)
  , Eq(dtime,encDTime)
  , Eq(from,encFrom)
  , Eq(to,encTo)]->
  [ SndS($D,Client,ack)]

rule H_4:
  let
    ack = <tknumber,date, 'ack', 'valid'>
  in
  [ State(Client,'4',<S,D,date,dtime,from,to,price
  ,tknumber>)
```

```
  , RcvS(D,Client,ack)
  ]
  --[End(Client,'ack','valid',tknumber,date)]->
  []

/****ENDOFRULES****/

restriction notSameRole:
  "All Client1 Client2 S1 S2 D1 D2 #i #j.
    Roles(Client1,S1,D1) @i & Roles(Client2,S2,D2) @j ==>
      not Client1 = S1
    & not Client1 = S2
    & not Client1 = D1
    & not Client1 = D2
    & not S1 = D1
    & not S1 = D2
    & Client1 = Client2
  "

restriction Equality:
  "All x y #i. Eq(x,y) @i ==> x = y"

restriction OnlyOnce:
  "All #i #j. OnlyOnce()@#i & OnlyOnce()@#j ==> #i = #j"

lemma functional: exists-trace
```

```
  "(Ex Client tkn date #m.
    End(Client,'ack','valid',tkn,date)@m)"

lemma auth: all-traces
  "All Client tn date #i.
    End(Client,'ack','valid',tn,date)@i ==>
      Ex Driver #j.
        ValidTicket(Driver,Client,'tn',tn,date)@j & j<i"

end

/****ENDOFMODEL****/
```

Listing A.5 The full code of the Ticket Inspection Ceremony